

Teil VI
Source-Code

Anhang I

Korpora – Code

I.1 Korpus-Verwaltung

Für das automatische Erstellen der `index.xml` Dateien wurden viele kleine Skripte geschrieben, von denen stellvertretend hier nur eines gezeigt wird.

I.1.1 koksUtil.tcl

Hilfsroutinen

```
1 # utility routines for the KOKS project
2 # Arno Erpenbeck, 06.01.2001
3
4 package provide koksUtil 1.0
5
6 namespace eval koksUtil {
7     namespace export openFile closeFile
8 }
9
10 # open a file specified by the name for reading
11 proc koksUtil::openFile {fnm {mode r}} {
12     if {[catch {set in [open $fnm $mode]} err]} {
13         puts stderr "Error opening index file: $err"
14         exit 1
15     }
16     return $in
17 }
18
19 # close the file specified by the handle
20 proc koksUtil::closeFile {handle} {
21     if {[catch {close $handle} err]} {
22         puts stderr "Error closing index file: $err"
23     }
24 }
```

I.1.2 denews-add.tcl

`index.xml` für DE-News anlegen

```
1 #!/bin/sh
2 # the next line restarts using tclsh \
3 exec tclsh8.3 "$0" "$@"
4
```

```

5 # $Header: /opt/cvs/bin/denews-add.tcl,v 1.1 2001/10/21 15:23:19 koks Exp $
6
7 # find all de-news text files and create index.xml
8 # KOKS project
9 # Arno Erpenbeck, 23.02.2001
10
11 # load packages
12 lappend auto_path /home/koks/share/tcl
13 package require koksUtil
14
15 set fnm "index.xml"
16 set verbose 0
17
18 # check command line args
19 if $argc {
20     if {![string compare "-v" [lindex $argv 0]]} {
21         set verbose 1
22     }
23 }
24
25 # traverse all subdirectories
26 foreach i [list 1996 1997 1998 1999 2000] {
27     foreach j [list 01 02 03 04 05 06 07 08 09 10 11 12] {
28         if $verbose {
29             puts -nonewline "$i/$j: "
30         }
31
32         # find text files in current dir
33         set files [glob -nocomplain $i/$j/*.txt]
34         if {[length $files]} {
35             if $verbose {
36                 puts "empty"
37             }
38             continue
39         }
40
41         # get day, month, year from filename
42         set dummy [split [lindex $files end] -]
43         set year [lindex $dummy 2]
44         set month [lindex $dummy 3]
45         set max [lindex [split [lindex $dummy 4] .] 0]
46
47         # create index.xml
48         set out [koksUtil::openFile $i/$j/$fnm w]
49
50         # write header
51         puts $out "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>"
52         puts $out "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">"
53         set foo [exec date "+%T %d.%m.%Y"]
54         puts $out "<!-- generated: $foo -->"
55         puts $out "<mapping>"
56         koksUtil::closeFile $out
57
58         # create xml entry for all files
59         for {set day 1} {$day <= $max} {incr day} {
60             # use "01" instead of "1"
61             if {$day <= 9} {
62                 set realday "0$day"
63             } else {
64                 set realday $day
65             }

```

```

66         exec idx-add.tcl -x -p -l de de-news-$year-$month-$realday.de.txt plain -p -l en de-news-$year-$month
67     }
68
69     # write footer
70     set out [koksUtil::openFile $i/$j/$fnm a]
71     puts $out "</mapping>"
72     koksUtil::closeFile $out
73
74     if $verbose {
75         puts "ok"
76     }
77 }
78 }

```

I.2 Korpus-Browser

GUI für Korpus-Navigation (s. 3.6.2)

```

1  #!/usr/local/bin/ruby
2
3  require "gtk"
4  require "tempfile"
5  require "xmlltreebuilder"
6  require "xmlltreevisitor"
7
8  # $Header: /opt/cvs/bin/kbrowser.rb,v 1.9 2001/08/13 16:21:43 koks-cvs Exp $
9
10 # browse index.xml files
11 # KOKS project
12 # Arno Erpenbeck, 30.07.2001
13
14 # TODO
15 # - progressbar for long operations
16 # - fix encoding problems
17 # - new xpms for toolbar
18 # - make preproc.py call possible
19
20 # define paths for external viewer
21 $ps_viewer = "/usr/X11R6/bin/gv"
22 $pdf_viewer = "/usr/X11R6/bin/acroread"
23 $html_viewer = "/usr/X11R6/bin/netscape"
24 $preproc_cmd = "/home/aerpenbe/bin/preproc.py"
25
26 # define xpms for tree display
27 $book_open_xpm = [
28     "16 16 4 1",
29     "      c None s None",
30     ".      c black",
31     "X      c #808080",
32     "o      c white",
33     "      ",
34     " ..      ",
35     ".Xo.    ... ",
36     ".Xoo.   ..oo. ",
37     ".Xooo.Xooo... ",
38     ".Xooo.oooo.X. ",
39     ".Xooo.Xooo.X. ",
40     ".Xooo.oooo.X. ",
41     ".Xooo.Xooo.X. ",
42     ".Xooo.oooo.X. ",
43     ".Xoo.Xoo..X. ",

```

```

44 " .Xo.O..ooX. ",
45 " .X..XXXXX. ",
46 " ..X..... ",
47 " .. ",
48 " ]
49
50 $book_closed_xpm = [
51 "16 16 6 1",
52 " c None s None",
53 ". c black",
54 "X c red",
55 "o c yellow",
56 "O c #808080",
57 "# c white",
58 " ",
59 " .. ",
60 " ..XX. ",
61 " ..XXXXX. ",
62 " ..XXXXXXXX. ",
63 ".ooXXXXXXXX. ",
64 ".ooXXXXXXXX. ",
65 ".X.ooXXXXXXXX. ",
66 ".XX.ooXXXXX.. ",
67 ".XX.ooXXX..#O ",
68 ".XX.oo..##00. ",
69 ".XX..##00.. ",
70 ".X.#00.. ",
71 " ..O.. ",
72 " .. ",
73 " ]
74
75 $mini_page_xpm = [
76 "16 16 4 1",
77 " c None s None",
78 ". c black",
79 "X c white",
80 "o c #808080",
81 " ",
82 " ..... ",
83 " .XXXXX.. ",
84 " .XoooX.X. ",
85 " .XXXXX.... ",
86 " .XooooXoo.o ",
87 " .XXXXXXXX.o ",
88 " .XooooooX.o ",
89 " .XXXXXXXX.o ",
90 " .XooooooX.o ",
91 " .XXXXXXXX.o ",
92 " .XooooooX.o ",
93 " .XXXXXXXX.o ",
94 " .....O ",
95 " oooooooooo ",
96 " ]
97
98 # define xpms for toolbar
99 $toolbar_sample_xpm = [
100 "20 19 5 1",
101 ". c None",
102 "# c #000000",
103 "i c #ffffff",
104 "s c #7f7f00",

```

```

105 "y c #ffff00",
106 ".....",
107 ".....",
108 ".....",
109 ".....###.....",
110 ".....#...#.#...",
111 ".....##.....",
112 "...###.....###.....",
113 "..#iy#####.....",
114 "..#iyiyiyiyi#.....",
115 "..#iyiyiyiyi#.....",
116 "..#iyiy#####.....",
117 "..#iy#ssssssss#...",
118 "..#iy#ssssssss#.....",
119 "..#y#ssssssss#.....",
120 "..##ssssssss#.....",
121 "..#####.....",
122 ".....",
123 ".....",
124 "....." ]
125
126 # create Pixmap object from xpm data
127 def new_pixmap(window, background)
128   pixmap, mask = Gdk::Pixmap::create_from_xpm_d(window, background, $toolbar_sample_xpm)
129   wpixmap = Gtk::Pixmap::new(pixmap, mask)
130 end
131
132 # setup colors and fonts
133 $black = Gdk::Color.new(0x0000, 0x0000, 0x0000)
134 $white = Gdk::Color.new(0xFFFF, 0xFFFF, 0xFFFF)
135 $blue = Gdk::Color.new(0x0000, 0x0000, 0xFFFF)
136 $green = Gdk::Color.new(0x0000, 0xCCCC, 0x0000)
137 $magenta = Gdk::Color.new(0xFFFF, 0x0000, 0xFFFF)
138 $red = Gdk::Color.new(0xFFFF, 0x0000, 0x0000)
139 $font = Gdk::Font.font_load("-adobe-courier-medium-r-normal--*-120-*-*-*-*")
140
141
142 # overload method for XML::DOM::NamedNodeMap
143 module XML
144 module DOM
145 class NamedNodeMap
146   def to_h
147     return {} if @nodes.nil?
148     hash = {}
149     @nodes.each do |k, v| hash[v.nodeName] = v.nodeValue end
150     hash
151   end
152   def to_s
153     return "" if @nodes.nil?
154     str = ""
155     @nodes.each do |k, v| str << v.to_s + ", " end
156     str.chop.chop
157   end
158 end
159 end
160 end
161
162 class MyWriter
163   def initialize(tempfile)
164     @tf = tempfile
165   end

```

```

166
167 def visit_Document(document)
168   @tf.print "<?xml version=\"1.0\"?>\n"
169   @tf.print "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">\n"
170   document.children_accept(self)
171 end
172
173 def visit_Element(element)
174   attrs = ""
175   element.attributes.each do |attr|
176     attrs += " " + attr.to_s
177   end
178   @tf.print "<#{element.nodeName}#{attrs}>"
179   element.children_accept(self);
180   @tf.print "</#{element.nodeName}>"
181 end
182
183 def visit_Text(text)
184   @tf.print text.nodeValue
185 end
186
187 def visit_Comment(text) end
188
189 end
190
191 =begin
192 # overload method for NQXML::NamedAttributes
193 module NQXML
194   class NamedAttributes
195     def attributesToString
196       return "" if @attrs.nil?
197       str = ""
198       @attrs.each do |key, val| str << "#{key}=\"#{val.to_s}\"", " end
199       return str.chop.chop # get rid of trailing ", "
200     end
201   end
202 end
203 =end
204
205
206 # split string (key-value list) into hash
207 class String
208   def to_h(delim=",")
209     tokens = self.split(delim)
210     hash = {}
211     tokens.each do |tok|
212       tok2 = tok.strip
213       ind = tok2.index("=")
214       key = tok2[0, ind]
215       val = tok2[ind+2..tok2.rindex("\")+1]
216       hash[key] = val
217     end
218     hash
219   end
220 end
221
222
223 # simple dialog with label and OK button
224 class MyDialog < Gtk::Dialog
225   def initialize(title, label, button="OK")
226     super()

```



```

227 border_width(0)
228 set_usize(200, 100)
229 signal_connect("destroy") do destroy end
230 set_title(title)
231 set_modal(true)
232
233 button = Gtk::Button::new(label)
234 button.flags |= Gtk::Widget::CAN_DEFAULT
235 button.signal_connect("clicked") do destroy end
236 action_area.pack_start(button)
237 button.grab_default
238
239 label = Gtk::Label::new(label)
240 label.set_padding(10, 10)
241 vbox.pack_start(label)
242
243 show_all
244 end
245 end
246
247
248 # somehow this is needed to destroy some widgets properly
249 module Destroyable
250   def destroy
251     super
252     @destroyed = true
253   end
254   def destroyed?
255     @destroyed
256   end
257 end
258
259
260 # show file selection dialog
261 class FileSelection < Gtk::FileSelection
262
263   def initialize()
264     super("Choose a korpus index file")
265     signal_connect("destroy") do destroy end
266     set_modal(true)
267
268     set_position(Gtk::WIN_POS_MOUSE)
269     hide_fileop_buttons
270
271     ok_button.signal_connect("clicked") do
272       yield(get_filename)
273       destroy
274     end
275     cancel_button.signal_connect("clicked") do
276       destroy
277     end
278   end
279
280 end
281
282
283 # notebook with customizable pages
284 class MyNotebook < Gtk::Notebook
285   include Destroyable
286
287   # needs parent window for pixmaps

```

```

288 def initialize(window)
289   super()
290   @destroyed = false
291   signal_connect("destroy") do destroy end
292   signal_connect("switch_page") do |widget, page, num_page|
293     unless destroyed?
294       page_switch(widget, page, num_page)
295     end
296   end
297   border_width(5)
298   set_tab_pos(Gtk::POS_TOP)
299   @book_open, @book_open_mask =
300     Gdk::Pixmap::create_from_xpm_d(window, nil, $book_open_xpm)
301   @book_closed, @book_closed_mask =
302     Gdk::Pixmap::create_from_xpm_d(window, nil, $book_closed_xpm)
303 end
304
305 private
306 def page_switch(widget, page, page_num)
307   oldpage = widget.cur_page
308   return if page == oldpage
309   page.tab_label.children[0].set(@book_open, @book_open_mask)
310   page.menu_label.children[0].set(@book_open, @book_open_mask)
311   if (oldpage)
312     oldpage.tab_label.children[0].set(@book_closed, @book_closed_mask)
313     oldpage.menu_label.children[0].set(@book_closed, @book_closed_mask)
314   end
315 end
316
317 public
318 def create_page(buffer, widget)
319   child = Gtk::Frame::new(buffer)
320   child.border_width(5)
321
322   vbox = Gtk::VBox::new(true, 0)
323   vbox.border_width(5)
324   child.add(vbox)
325
326   hbox = Gtk::HBox::new(true, 0)
327   vbox.pack_start(hbox, false, true, 5)
328
329   hbox.pack_start(widget)
330   child.show_all
331
332   label_box = Gtk::HBox::new(false, 0)
333   pixwid = Gdk::Pixmap::new(@book_closed, @book_closed_mask)
334   label_box.pack_start(pixwid, false, true, 0)
335   pixwid.set_padding(3, 1)
336   label = Gtk::Label::new(buffer)
337   label_box.pack_start(label, false, true, 0)
338   label_box.show_all
339
340   menu_box = Gtk::HBox::new(false, 0)
341   pixwid = Gdk::Pixmap::new(@book_closed, @book_closed_mask)
342   menu_box.pack_start(pixwid, false, true, 0)
343   pixwid.set_padding(3, 1)
344   label = Gtk::Label::new(buffer)
345   menu_box.pack_start(label, false, true, 0)
346   menu_box.show_all
347
348   append_page_menu(child, label_box, menu_box)

```

```

349     end
350
351 end
352
353
354 # show contents of file in scrollable window
355 class TextWindow < Gtk::Window
356   def initialize(filename, colors=false)
357     super()
358     set_title("KBrowser - " + File.basename(filename))
359     set_usize(500, 500)
360     set_policy(true, true, false)
361     signal_connect("destroy") do destroy end
362
363     # create surrounding boxes
364     box1 = Gtk::VBox.new(false, 0)
365     add(box1)
366     box1.show
367
368     box2 = Gtk::VBox.new(false, 10)
369     box2.border_width(10)
370     box1.pack_start(box2, true, true, 0)
371     box2.show
372
373     scrolled_window = Gtk::ScrolledWindow.new(nil, nil)
374     box2.pack_start(scrolled_window, true, true, 0)
375     scrolled_window.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS)
376     scrolled_window.show
377
378     # create Text widget
379     text = Gtk::Text.new(nil, nil)
380     text.set_editable(false)
381     scrolled_window.add(text)
382     text.grab_focus
383     text.show
384     text.freeze
385
386     # load content of give file into text widget
387     begin
388       buffer = nil
389       File.open(filename, "r") do |infile|
390         buffer = infile.read
391       end
392       if colors
393         # split each line at color tags and segment/sentence boundaries
394         buffer.each_line do |l|
395           tok = l.split('\t')
396           if tok.size == 3
397             text.insert($font, $black, $white, tok[0]+\t")
398             text.insert($font, $blue, $white, tok[1]+\t")
399             if tok[2] =~ /^</
400               text.insert($font, $red, $white, tok[2])
401             elsif tok[2] =~ /^@/
402               text.insert($font, $magenta, $white, tok[2])
403             else
404               text.insert($font, $black, $white, tok[2])
405             end
406           elsif l =~ /^</
407             text.insert($font, $green, $white, l)
408           else
409             text.insert($font, nil, nil, l)

```

```

410         end
411     end
412     else
413         text.insert($font, nil, nil, buffer)
414     end
415     text.thaw
416 rescue
417     STDERR.puts "Hoppla: #{!}"
418 end
419 end
420
421 end
422
423 # show contents of two files in scrollable windows
424 class DoubleWindow < Gtk::Window
425     def initialize(filename)
426         super()
427         set_title("KBrowser - " + File.basename(filename))
428         #set_usize(600, 500)
429         set_policy(true, true, false)
430         signal_connect("destroy") do destroy end
431         signal_connect("delete_event") do false end
432         @filename = [2]
433         @filename[0] = filename
434         if filename =~ /de/
435             @filename[1] = filename.sub(/de/, "en")
436         else
437             @filename[1] = filename.sub(/en/, "de")
438         end
439         init
440     end
441
442     private
443     def init
444         # create surrounding boxes
445         box1 = Gtk::VBox.new(false, 0)
446         add(box1)
447         box1.show
448
449         #box2 = Gtk::HBox.new(false, 10)
450         box2 = Gtk::HPaned.new()
451         box2.border_width(10)
452         box1.pack_start(box2, true, true, 0)
453         box2.show
454
455         # create Text widget
456         text = [2]
457         0.upto(1) do |i|
458             scrolled_window = Gtk::ScrolledWindow.new(nil, nil)
459             #box2.pack_start(scrolled_window, true, true, 0)
460             if i == 0
461                 box2.add1(scrolled_window)
462             else
463                 box2.add2(scrolled_window)
464             end
465             # Text widgets currently do not support horizontal scrollbars
466             scrolled_window.set_policy(Gtk::POLICY_AUTOMATIC, Gtk::POLICY_ALWAYS)
467         =begin
468             scrolled_window.signal_connect("motion_notify_event") do |w, c|
469                 print "x=#{c.x}, y=#{c.y}, x_root=#{c.x_root}, y_root=#{c.y_root}\n"
470             end

```

```

471 =end
472   scrolled_window.show
473
474   text[i] = Gtk::Text.new(nil, nil)
475   text[i].set_editable(false)
476   text[i].set_usize(300, 400)
477   scrolled_window.add(text[i])
478   text[i].grab_focus
479   text[i].show
480   text[i].freeze
481   read_file(text[i], i)
482   text[i].thaw
483   end
484 end
485
486 private
487 def read_file(widget, nr)
488   # load content of give file into text widget
489   begin
490     buffer = nil
491     File.open(@filename[nr], "r") do |infile|
492       buffer = infile.read
493     end
494     buffer.each_line do |l|
495       tok = l.split('\t')
496       if tok.size == 3
497         widget.insert($font, nil, nil, tok[0]+" ")
498       elsif l =~ /<SATZ>/
499         widget.insert($font, nil, nil, "\n")
500       elsif l =~ /<segmentgrenze>/
501         widget.insert($font, nil, nil, "\n\n")
502       end
503     end
504   rescue
505     STDERR.puts "Hoppla: #{!}"
506   end
507 end
508
509 end
510
511
512 # shows XML tree in a Gtk::CTree
513 class MyTree < Gtk::CTree
514
515   attr_accessor :verbose
516
517   def initialize(cols, n, window)
518     super(cols, n)
519
520     set_column_auto_resize(0, true);
521     set_column_width(1, 200);
522     set_selection_mode(Gtk::SELECTION_SINGLE);
523     @line_style = Gtk::CTree::LINES_DOTTED;
524     @node_style = Gtk::Style::new()
525     @node_style.set_base(Gtk::STATE_NORMAL, 0, 45000, 55000)
526
527     signal_connect("click_column") do |w,c| click_column(c) end
528
529     @pixmap1, @mask1 = Gdk::Pixmap::create_from_xpm_d(window, style.white, $book_closed_xpm)
530     @pixmap2, @mask2 = Gdk::Pixmap::create_from_xpm_d(window, style.white, $book_open_xpm)
531     @pixmap3, @mask3 = Gdk::Pixmap::create_from_xpm_d(window, style.white, $mini_page_xpm)

```

```

532
533     set_usize(0, 300)
534     @filelist = [] # keeps all filenames collected from xml file
535     @verbose = false
536 end
537
538 # rebuild tree if data has changed
539 public
540 def rebuild_tree(filename)
541     @filename = filename
542     freeze
543     clear
544     @filelist.clear
545     begin
546         STDERR.print("Parsing XML file... (#{Time.now})") if @verbose
547         parser = XML::SimpleTreeBuilder.new(0, "ISO-8859-1")
548         doc = parser.parse(File.new(@filename))
549         STDERR.puts(" ...done (#{Time.now})") if @verbose
550     rescue
551         STDERR.puts "Hoppla: #{!}"
552         thaw
553         return
554     end
555     @root = doc.documentElement
556     STDERR.puts("Inserting parent") if @verbose
557     parent = insert_node(nil, nil, [@root.nodeName, @root.attributes.to_s], 5,
558         @pixmap1, @mask1, @pixmap2, @mask2, false, true)
559
560     node_set_row_data(parent, @node_style)
561     STDERR.print("Starting recursive build... (#{Time.now})") if @verbose
562     build_recursive(@root, parent)
563     STDERR.puts(" ...done (#{Time.now})") if @verbose
564     thaw
565     parser.done # clear up remaining garbage
566     @filelist # return filelist, so that CList can display it
567 end
568
569 # recursively walk through document model and fill tree/list with data
570 private
571 def build_recursive(node, parent)
572     sibling = nil
573     node.childNodes.each do |child|
574         case child.nodeType
575             when XML::DOM::Node::TEXT_NODE
576                 str = child.nodeValue.strip
577                 if str.length > 0
578                     if child.parentNode.nodeName == "filename"
579                         @filelist << [str, child.parentNode.attributes.to_h["format"]]
580                     end
581                     sibling = insert_node(parent, sibling,
582                         [",", str],
583                         5, @pixmap3, @mask3, nil, nil, true, false)
584                 end
585             when XML::DOM::Node::COMMENT_NODE
586                 else
587                     sibling = insert_node(parent, sibling,
588                         [child.nodeName, child.attributes.to_s],
589                         5, @pixmap1, @mask1, @pixmap2, @mask2, false, false)
590                     node_set_row_data(sibling, @node_style)
591                     build_recursive(child, sibling)
592                 end

```

```

593     end
594 end
595
596 =begin
597 def old_build_recursive(node, parent)
598   # old version for NQXML
599   sibling = nil
600   entity = node.entity
601   node.children.each do |child|
602     entity = child.entity
603     if entity.class == NQXML::Text
604       str = entity.to_s.strip
605       if str.length > 0
606         if child.parent.entity.name == "filename"
607           @filelist << [entity.to_s, child.parent.entity.attributesToString.to_h["format"]]
608         end
609         sibling = insert_node(parent, sibling,
610           [ "", entity.to_s ],
611           5, @pixmap3, @mask3, nil, nil, true, false)
612       end
613     elsif entity.class == NQXML::Tag
614       sibling = insert_node(parent, sibling,
615         [ child.entity.name, child.entity.attributesToString ],
616         5, @pixmap1, @mask1, @pixmap2, @mask2, false, false)
617       node_set_row_data(sibling, @node_style)
618       build_recursive(child, sibling)
619     end
620   end
621 end
622 =end
623
624 public
625 def subtree(id)
626   tf = nil
627   @root.childNodes.each do |child|
628     if child.nodeName == "document" && child.attributes.to_h["id"] == id
629       temproot = XML::DOM::Element.new("mapping", nil, child.clone)
630       tempdoc = XML::DOM::Document.new(temproot)
631       tf = Tempfile.new("preproc", File.dirname(@filename))
632       $STDERR.puts "Generating #{tf.path}"
633       tempdoc.accept(MyWriter.new(tf))
634       tf.close
635       break
636     end
637   end
638   tf
639 end
640
641 # user clicked on column title in tree page
642 private
643 def click_column(column)
644   if column == @ctree.sort_column then
645     if sort_type == Gtk::SORT_ASCENDING then
646       sort_type = Gtk::SORT_DESCENDING
647     else
648       sort_type = Gtk::SORT_ASCENDING
649     end
650   else
651     sort_column = column
652   end
653   sort_recursive(nil)

```

```

654     end
655
656 end
657
658
659 # shows filenames in a Gtk::CList
660 class MyList < Gtk::CList
661
662     def initialize(cols)
663         super(cols)
664         set_selection_mode(Gtk::SELECTION_SINGLE);
665         column_titles_passive
666         column_titles_show
667         set_column_auto_resize(0, true)
668         set_column_resizeable(1, false)
669     end
670
671     # rebuild list if data has changed
672     public
673     def rebuild_list(filelist)
674         if filelist.size > 0
675             freeze
676             clear
677             filelist.each do |v| append([v[0],v[1]]) end
678             thaw
679         end
680     end
681
682 end
683
684 # shows radio buttons to select options for preproc.py
685 class CheckBox < Gtk::Window
686     def initialize(title)
687         super(Gtk::WINDOW_TOPLEVEL)
688         set_title(title)
689         @choice = ["-i kbrowser", "-c"]
690         signal_connect("destroy") do destroy end
691         signal_connect("delete_event") do false end
692         init
693     end
694
695     private
696     def init
697         box1 = Gtk::VBox::new(false, 0)
698         box1.border_width(5)
699         add(box1)
700         box1.show
701
702         label = Gtk::Label::new("What do you want preproc to do?")
703         box1.pack_start(label, true, true, 0)
704         label.show
705
706         frame = Gtk::Frame::new("Task")
707         frame.border_width(5)
708         box1.pack_start(frame, true, true, 0)
709         frame.show
710
711         box2 = Gtk::VBox::new(false, 10)
712         box2.border_width(10)
713         frame.add(box2)
714         box2.show

```



```

715
716 button = Gtk::RadioButton::new(nil, "Convert")
717 button.signal_connect("toggled") do @choice = ["-i kbrowser", "-c"] end
718 box2.pack_start(button, true, true, 0)
719 button.show
720
721 button = Gtk::RadioButton::new(button.group, "Tag")
722 button.signal_connect("toggled") do @choice = ["-i kbrowser", "-n", "-a"] end
723 box2.pack_start(button, true, true, 0)
724 button.show
725
726 button = Gtk::RadioButton::new(button.group, "Align")
727 button.signal_connect("toggled") do @choice = ["-i kbrowser", "-n"] end
728 box2.pack_start(button, true, true, 0)
729 button.show
730
731 button = Gtk::RadioButton::new(button.group, "Enter DB")
732 button.signal_connect("toggled") do @choice = ["-i kbrowser"] end
733 box2.pack_start(button, true, true, 0)
734 button.show
735
736 separator = Gtk::HSeparator::new()
737 box1.pack_start(separator, false, true, 0)
738 separator.show
739
740 box2 = Gtk::HBox::new(false, 10)
741 box2.border_width(10)
742 box1.pack_start(box2, false, true, 0)
743 box2.show
744
745 button = Gtk::Button::new("Cancel")
746 button.signal_connect("clicked") do destroy end
747 box2.pack_start(button, true, true, 0)
748 button.show
749
750 button = Gtk::Button::new("Run")
751 button.signal_connect("clicked") do
752   yield(@choice)
753   destroy
754 end
755 box2.pack_start(button, true, true, 0)
756 button.set_flags(Gtk::Widget::CAN_DEFAULT)
757 button.grab_default
758 button.show
759
760 end
761 end
762
763 # the main window which does all the work
764 class MainWindow < Gtk::Window
765
766   def initialize(title, verbose=false)
767     super(Gtk::WINDOW_TOPLEVEL)
768     @filename = nil
769     @title = title
770     @verbose = verbose
771     set_title(@title)
772     set_policy(true, true, false);
773     set_name("main window")
774     set_usize(600, 400)
775     set_uposition(40, 40)

```

```

776
777     signal_connect("destroy") do Gtk.main_quit end
778     signal_connect("delete_event") do false end
779
780     init
781 end
782
783 # set filename, triggers rebuilding of tree/list
784 def set_filename(fnm)
785     @filename = fnm
786     @dirname = File.dirname(fnm)
787     set_title(@title + " - " + fnm)
788     filelist = @ctree.rebuild_tree(fnm)
789     @clist.rebuild_list(filelist)
790     message("Loaded index file " + @filename)
791 end
792
793 # initialize GUI
794 private
795 def init
796     @box = Gtk::VBox::new(false, 0)
797     add(@box)
798
799     create_menu
800     create_toolbar
801     create_notebook
802     create_statusbar
803
804     @box.show
805 end
806
807 # create a statusbar for useless messages
808 private
809 def create_statusbar
810     @statusbar = Gtk::Statusbar::new()
811     @box.pack_end(@statusbar, false, false, 0)
812     @main_id = @statusbar.get_context_id("Main")
813     #@toolbar_id = @statusbar.get_context_id("Toolbar")
814     @statusbar.push(@main_id, "Welcome to KBrowser!")
815     @statusbar.show
816 end
817
818 # create a menu bar
819 private
820 def create_menu
821     menubar = Gtk::MenuBar::new()
822     menu = Gtk::Menu::new()
823
824     # create File menu
825     menuitem = Gtk::MenuItem::new("File")
826     submenu = Gtk::Menu::new()
827     smenuitem = Gtk::MenuItem::new("Open")
828     submenu.append(smenuitem)
829     smenuitem.signal_connect("activate") do
830         f = FileSelection.new {|f| set_filename(f)}
831         f.show
832     end
833     smenuitem = Gtk::MenuItem::new("Quit")
834     smenuitem.signal_connect("activate") do Gtk.main_quit end
835     submenu.append(smenuitem)
836

```

```

837 # connect to menubar
838 menuitem.set_submenu(submenu)
839 menubar.append(menuitem)
840
841 # create Help menu
842 menuitem = Gtk::MenuItem::new("Help")
843 submenu = Gtk::Menu::new()
844 smenuitem = Gtk::MenuItem::new("About")
845 smenuitem.signal_connect("activate") do
846   MyDialog.new("About KBrowser", "KBrowser rulez!")
847 end
848 submenu.append(smenuitem)
849
850 # connect to menubar
851 menuitem.set_submenu(submenu)
852 menuitem.right_justify
853 menubar.append(menuitem)
854
855 menubar.show_all
856 @box.pack_start(menubar, false, true, 0)
857 end
858
859 # create a toolbar with nice buttons
860 private
861 def create_toolbar
862   tbox = Gtk::HBox::new(false, 5)
863   @box.pack_start(tbox, false, false, 0)
864   tbox.show
865   toolbar = Gtk::Toolbar::new(Gtk::ORIENTATION_HORIZONTAL, Gtk::TOOLBAR_BOTH)
866   toolbar.set_button_relief(Gtk::RELIEF_NONE)
867   toolbar.set_space_style(Gtk::Toolbar::SPACE_LINE)
868   realize
869   toolbar.append_item("Original", "View original file in korpus",
870     "Toolbar/Original",
871     new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
872     nil) do
873     show_file(5, false)
874   end
875   toolbar.append_item("Ascii", "View ASCII file in korpus",
876     "Toolbar/Ascii",
877     new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
878     nil) do |w|
879     show_file(1, false)
880   end
881   toolbar.append_item("Tagged", "View tagged file in korpus",
882     "Toolbar/Tagged",
883     new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
884     nil) do
885     show_file(2, true)
886   end
887   toolbar.append_item("Aligned", "View aligned file in korpus",
888     "Toolbar/Aligned",
889     new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
890     nil) do
891     show_file(3, true)
892   end
893   toolbar.append_item("Both", "View both aligned files in korpus",
894     "Toolbar/Both",
895     new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
896     nil) do
897     show_file(4, true)

```

```

898     end
899     toolbar.append_space
900     toolbar.append_item("Preproc", "Run preproc.py on selected document",
901         "Toolbar/Preproc",
902         new_pixmap(window, style.bg(Gtk::STATE_NORMAL)),
903         nil) do
904         preproc
905     end
906     tbox.add(toolbar)
907     toolbar.show
908 end
909
910 # construct local subtree
911 private
912 def preproc
913     return if @notebook.get_current_page == 1
914     @ctree.each_selection do |s|
915         if @ctree.node_get_pixtext(s, 0)[0] == "document"
916             id = @ctree.node_get_text(s, 1).to_h["id"]
917             next unless id
918             tf = @ctree.subtree(id)
919             cb = CheckBox.new("Select task") do |c|
920                 @statusbar.push(@main_id, "Running preproc.py")
921                 args = c << tf.path
922                 launch_external($preproc_cmd, args)
923             end
924             cb.show
925         end
926     end
927 end
928
929 # get filename of selected entry from tree/list
930 private
931 def get_selected
932     fnm = ""
933     format = ""
934     case @notebook.get_current_page
935     when 0
936         @ctree.each_selection do |s|
937             if s.leaf? && @ctree.node_get_pixtext(s.parent, 0)[0] == "filename"
938                 # only works if filename selected
939                 fnm = @ctree.node_get_text(s, 1)
940                 fnm = yield(fnm) if block_given? # get correct files suffix
941                 format = @ctree.node_get_text(s.parent, 1).to_h["format"]
942             end
943         end
944     when 1
945         @clist.each_selection do |s|
946             fnm = @clist.get_text(s, 0)
947             fnm = yield(fnm) if block_given?
948             format = @clist.get_text(s, 1)
949         end
950     end
951     if fnm.length > 0
952         return [fnm, format]
953     else
954         return nil
955     end
956 end
957
958 # show original/ascii/tagged/aligned korpus file

```

```

959 private
960 def show_file(type, color=false)
961   case type
962     when 1
963       fnm, format = get_selected {|f| f[0, f.rindex(".") + 1] + "asc2"}
964     when 2
965       fnm, format = get_selected {|f| f[0, f.rindex(".") + 1] + "tag"}
966     when 3,4
967       fnm, format = get_selected {|f| f[0, f.rindex(".") + 1] + "tag.al"}
968     when 5
969       fnm, format = get_selected
970   end
971   return unless fnm
972   cfnm = @dirname + File::SEPARATOR + fnm
973   if FileTest.exist?(cfnm)
974     succesful = true
975     if type == 5
976       # launch external viewer
977       if format == "plain" || format == "tagged" || format == "sgml"
978         TextWindow.new(cfnm).show
979       elsif format == "ps"
980         launch_external($ps_viewer, cfnm)
981       elsif format == "pdf"
982         launch_external($pdf_viewer, cfnm)
983       elsif format == "html"
984         launch_external($html_viewer, cfnm)
985       elsif format == "unknown" || format == "doc"
986         succesful = false
987       end
988     elsif type == 4
989       DoubleWindow.new(cfnm).show
990     else
991       # open text window
992       TextWindow.new(cfnm, color).show
993     end
994     if succesful
995       message("Launched viewer for " + fnm)
996     else
997       message("Don't know what to do with file " + fnm)
998     end
999   else
1000     message("File not found: " + cfnm)
1001   end
1002 end
1003
1004 # launch external viewer and trap signal of its death
1005 private
1006 def launch_external(cmd, args)
1007   trap("CLD") do
1008     pid = Process.wait
1009     message("Child pid #{pid}: terminated")
1010   end
1011   exec(cmd, *args) if fork == nil
1012 end
1013
1014 private
1015 def create_notebook
1016   @notebook = MyNotebook.new(window)
1017   @box.pack_start(@notebook, true, true, 0)
1018   @notebook.create_page("Tree View", create_tree)
1019   @notebook.create_page("List View", create_list)

```

```

1020     @notebook.show
1021 end
1022
1023 private
1024 def create_list
1025     vbox = Gtk::VBox::new()
1026     scrolled_win = Gtk::ScrolledWindow::new()
1027     scrolled_win.set_policy(Gtk::POLICY_NEVER, Gtk::POLICY_ALWAYS)
1028     vbox.pack_start(scrolled_win)
1029
1030     @clist = MyList.new(["File", "Format"])
1031     @clist.signal_connect("select_row") do message("") end
1032     scrolled_win.add(@clist)
1033
1034     vbox
1035 end
1036
1037 private
1038 def create_tree
1039     vbox = Gtk::VBox::new()
1040     scrolled_win = Gtk::ScrolledWindow::new()
1041     scrolled_win.border_width(5)
1042     scrolled_win.set_policy(Gtk::POLICY_AUTOMATIC, Gtk::POLICY_ALWAYS)
1043     vbox.pack_start(scrolled_win)
1044
1045     @ctree = MyTree.new(["Tree", "Info"], 0, window)
1046     @ctree.verbose = @verbose
1047     @ctree.signal_connect_after("button_press_event") do message("") end
1048     scrolled_win.add(@ctree)
1049
1050     hbox = Gtk::HBox::new(false, 5)
1051     vbox.pack_start(hbox, false, false, 0)
1052
1053     tooltips = Gtk::Tooltips::new()
1054
1055     button = Gtk::Button::new("Expand All")
1056     hbox.pack_start(button)
1057     button.signal_connect("clicked") do
1058         @ctree.expand_recursive(nil)
1059         message("Expanded XML Tree")
1060     end
1061     tooltips.set_tip(button, "Expand all nodes of the tree", "Buttons/Expand")
1062
1063     button = Gtk::Button::new("Collapse All")
1064     hbox.pack_start(button)
1065     button.signal_connect("clicked") do
1066         @ctree.collapse_recursive(nil)
1067         message("Collapsed XML Tree")
1068     end
1069     tooltips.set_tip(button, "Collapse all nodes of the tree", "Buttons/Collapse")
1070
1071     vbox
1072 end
1073
1074 # set new message in status bar
1075 def message(text)
1076     @statusbar.pop(@main_id) unless @statusbar.messages.empty?
1077     @statusbar.push(@main_id, text)
1078 end
1079
1080 end

```

```

1081
1082
1083 def main
1084   window = MainWindow.new("KBrowser", false)
1085   window.show()
1086   if ARGV.length == 1
1087     window.set_filename(ARGV[0])
1088   end
1089   Gtk::main()
1090 end
1091
1092 main

```

I.3 XML Tools

I.3.1 mapping.dtd

DTD für index.xml (s. C.2.1)

```

1 <?xml encoding="ISO-8859-1"?>
2
3 <!--
4     DTD for file name mapping of German and English corpus documents
5     KOKS project
6     Last change: Arno Erpenbeck, 20.07.2001
7 -->
8
9 <!-- $Header: /opt/cvs/share/mapping.dtd,v 1.10 2001/10/25 11:51:18 koks-cvs Exp $ -->
10
11 <!-- some information remains unknown -->
12 <!ENTITY % unknown.att "unknown">
13
14 <!-- flag indicating whether a document can be user for further processing -->
15 <!ENTITY % use.att "yes|no">
16
17 <!-- document kind is either lexicon, phrase, or text -->
18 <!ENTITY % kind.att "lexicon|phrase|text">
19
20 <!-- documents can be either German (de) or English (en)
21     or both languages in one file (bi) -->
22 <!ENTITY % lang.att "de|en|bi">
23
24 <!-- each part can use its own tag set -->
25 <!ENTITY % set.att "ims|lql">
26
27 <!-- allowed file formats, list should be extended as needed -->
28 <!ENTITY % format.att "plain|html|sgml|pdf|ps|doc|tagged|%unknown.att;">
29
30 <!-- recognized encodings for text files -->
31 <!ENTITY % enc.att "iso-8859-1|%unknown.att;">
32
33 <!-- allowed types for document source -->
34 <!ENTITY % type.att "url|cdrom|%unknown.att;">
35
36 <!-- allowed methods for document checksums -->
37 <!ENTITY % method.att "cksum|md5|sha1">
38
39 <!ELEMENT mapping (document+)>
40
41 <!-- each document consists of one or more parts and additional info -->
42 <!ELEMENT document (part+,info?)>

```

```

43 <!ATTLIST document
44         id ID #IMPLIED
45         use (%use.att;) #IMPLIED
46         kind (%kind.att;) #IMPLIED>
47
48 <!-- each part consists of one or more files in a specified language -->
49 <!ELEMENT part (file+)>
50 <!ATTLIST part
51         lang (%lang.att;) #REQUIRED
52         tagset (%set.att;) #IMPLIED>
53
54 <!-- a file is identified by its name and an optional checksum -->
55 <!ELEMENT file (filename,checksum?)>
56
57 <!-- filename contains the relative path to the described file,
58      also there is an optional encoding tag -->
59 <!ELEMENT filename (#PCDATA)>
60 <!ATTLIST filename
61         format (%format.att;) #REQUIRED
62         enc (%enc.att;) #IMPLIED>
63
64 <!-- checksum can be used to uniquely identify documents -->
65 <!ELEMENT checksum (#PCDATA)>
66 <!ATTLIST checksum method (%method.att;) #REQUIRED>
67
68 <!-- the info tag can be further refined, e.g. for source information -->
69 <!ELEMENT info (author,comment,date,source)>
70
71 <!ELEMENT author (#PCDATA)>
72 <!ELEMENT comment (#PCDATA)>
73 <!ELEMENT date (#PCDATA)>
74
75 <!-- type of source can remain unspecified -->
76 <!ELEMENT source (#PCDATA)>
77 <!ATTLIST source type (%type.att;) #IMPLIED>
78

```

I.3.2 meta.dtd

DTD für metaindex.xml (s. C.2.1)

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2
3 <!--
4     DTD for meta index listing all index.xml files in corpus directory
5     KOKS project
6     Last change: Arno Erpenbeck, 19.07.2001
7 -->
8
9 <!-- $Header: -->
10
11 <!ELEMENT meta-index (korpus+)>
12
13 <!ELEMENT korpus (file*)>
14 <!ATTLIST korpus
15         name CDATA #REQUIRED>
16
17 <!ELEMENT file (#PCDATA)>

```

I.3.3 idx-ls.tcl

Listet Inhalt von index.xml auf (s. C.1.3)


```

1 #!/bin/sh
2 # the next line restarts using tclsh \
3 exec tclsh8.3 "$0" "$@"
4
5 # $Header: /opt/cvs/bin/idx-ls.tcl,v 1.1 2001/10/21 15:20:24 koks Exp $
6
7 # list contents of an index.xml file
8 # KOKS project
9 # Arno Erpenbeck, 03.04.2001
10
11 # make sure koksUtil can be found
12 lappend auto_path /home/koks/share/tcl
13
14 # load packages
15 package require xml
16 package require koksUtil
17
18 # set up global variables
19 set version "0.4 - 03.04.2001"
20 set fnm "index.xml"
21 set count(total) 0
22 set count(part) 0
23 set flag(filter) ""
24 set flag(form) 0
25 set flag(lang) 0
26 set flag(file) 0
27 set flag(skip) 0
28 set flag(stat) 0
29 set flag(terse) 0
30 set flag(changed) 0
31 set flag(checksum) 0
32 set line(checkfile) ""
33 set line(content) ""
34
35 # -----
36
37 # handle opening tags
38 proc HandleStart {name attlist} {
39     global flag line count
40     switch -exact -- $name {
41         document {
42             # only print newline if more than one document
43             if $count(part) {puts ""}
44             incr count(total)
45         }
46         part {
47             set l [lindex $attlist 1]
48             if $flag(lang) {
49                 # only print parts of matching language
50                 if {[string compare $l $flag(filter)]} {
51                     # current lang matches specified
52                     if !$flag(terse) {
53                         set line(content) "$l: "
54                     }
55                 } else {
56                     # skip this part
57                     set flag(skip) 1
58                 }
59             } elseif !$flag(terse) {
60                 set line(content) "$l: "
61             } else {

```

```

62         set line(content) ""
63     }
64 }
65 checksum {
66     # if "-c" flag, test for checksum (currently only md5)
67     if {$flag(changed) && ![string compare [lindex $attlist 1] "md5"]} {
68         set flag(checksum) 1
69     }
70 }
71 filename {
72     if !$flag(skip) {
73         # only print if part not skipped
74         if $flag(form) {
75             append line(content) "([lindex $attlist 1]) "
76         }
77         set flag(file) 1
78     }
79 }
80 }
81 }
82
83 # handle closing tags
84 proc HandleEnd {name} {
85     global flag line count
86     switch -exact -- $name {
87         mapping {
88             if $flag(stat) {
89                 puts "--> $count(total) documents ($count(part) parts)"
90             }
91         }
92         part {
93             if !$flag(skip) {
94                 puts $line(content)
95                 incr count(part)
96             }
97             set flag(skip) 0
98         }
99         checksum {
100             # clear flags
101             set flag(checksum) 0
102             set line(checkfile) ""
103         }
104         filename {
105             set flag(file) 0
106         }
107     }
108 }
109
110 # handle text data between tags
111 proc HandleText {data} {
112     global flag line
113     if $flag(file) {
114         # we are inside a <filename> tag
115         #set line(checkfile) [string trim $data]
116         if !$flag(changed) {
117             # print out every filename
118             append line(content) [string trim $data]
119         } else {
120             # keep filename, only print if changed
121             set line(checkfile) [string trim $data]
122         }

```

```

123 } elseif $flag(checksum) {
124     # we are inside a <checksum> tag
125     # compute md5 checksum and complain if changed
126     set cfnm $line(checkfile)
127     if {[catch {set result [lindex [split [exec md5 $line(checkfile)]] 3]} err]} {
128         puts stderr "Error: cannot generate checksum using md5!"
129     }
130     if !$flag(changed) {
131         append line(content) $line(checkfile)
132     } elseif {![string compare [string trim $data] $result]} {
133         # checksum ok, skip entry
134         set flag(skip) 1
135     } else {
136         # checksum changed, print out filename
137         append line(content) $line(checkfile)
138     }
139 }
140 }
141
142
143 # check if file obeys to correct DTD
144 proc HandleDocType {docelement publicID systemID internalDTD} {
145     if {[string compare $systemID "/home/koks/share/mapping.dtd"]} {
146         puts stderr "index file is of wrong DTD: $systemID"
147         exit 2
148     }
149 }
150
151 # print help message
152 proc Help {} {
153     puts "Usage: idx-ls.tcl \[-c\] \[-f\] \[-s\] \[-t\] \[-l <lang>\] \[index-file\]"
154     puts "-c: compute checksum and only list changed files"
155     puts "-f: print out file format of each file"
156     puts "-l <lang>: only list entries of language <lang> (de, en, bi)"
157     puts "-s: print some statistics in the end"
158     puts "-t: terse mode, only list file names"
159     exit
160 }
161
162 # print version info
163 proc Version {} {
164     global version
165     puts $version
166     exit
167 }
168
169 # -----
170
171 # check command line args
172 foreach arg $argv {
173     switch -exact -- $arg {
174         -h {Help}
175         -v {Version}
176         -c {set flag(changed) 1}
177         -f {set flag(form) 1}
178         -l {set flag(lang) 1}
179         -s {set flag(stat) 1}
180         -t {set flag(terse) 1}
181         de {if $flag(lang) {set flag(filter) $arg}}
182         en {if $flag(lang) {set flag(filter) $arg}}
183         bi {if $flag(lang) {set flag(filter) $arg}}

```

```

184     default {set fnm $arg}
185   }
186 }
187
188 # open index file
189 set in [koksUtil::openFile $fnm]
190
191 # create a new xml parser
192 set p [xml::parser]
193
194 # configure parser
195 $p configure -doctypecommand HandleDocType
196 $p configure -elementstartcommand HandleStart
197 $p configure -elementendcommand HandleEnd
198 $p configure -characterdatacommand HandleText
199
200 # parse document
201 $p parse [read $in]
202
203 # close file
204 koksUtil::closeFile $in

```

I.3.4 idx-add.tcl

Generiert index.xml (s. C.1.3)

```

1  #!/bin/sh
2  # the next line restarts using tclsh \
3  exec tclsh8.3 "$0" "$@"
4
5  # $Header: /opt/cvs/bin/idx-add.tcl,v 1.1 2001/10/21 15:20:24 koks Exp $
6
7  # create an index.xml file
8  # KOKS project
9  # Arno Erpenbeck, 16.05.2001
10
11 # set up global variables
12 set version "0.3 - 16.05.2001"
13 set valid(format) [list plain html sgml pdf ps doc unknown]
14 set valid(type) [list url cdrom unknown]
15 set valid(lang) [list de en bi]
16 set valid(use) [list yes no]
17 set count 0
18 set small 0
19 set files(1) [list]
20 set files(md5) 0
21 set lang(1) ""
22 set useFlag ""
23 set info(author) ""
24 set info(comment) ""
25 set info(date) ""
26 set info(source) ""
27 set info(type) ""
28
29 # -----
30
31 # print xml file header
32 proc Header {} {
33     global useFlag
34     puts "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>"
35     puts "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">"

```

```

36 puts "<mapping>"
37 if [string length $useFlag] {
38     puts "<document use=\"\$useFlag\">"
39 } else {
40     puts "<document>"
41 }
42 }
43
44 # print small xml file header
45 proc SmallHeader {} {
46     global useFlag
47     if [string length $useFlag] {
48         puts "<document use=\"\$useFlag\">"
49     } else {
50         puts "<document>"
51     }
52 }
53
54 # print xml file footer
55 proc Footer {} {
56     puts "</document>"
57     puts "</mapping>"
58 }
59
60 # print small xml file footer
61 proc SmallFooter {} {
62     puts "</document>"
63 }
64
65 # write the different parts
66 proc WriteParts {count} {
67     global valid files lang
68     for {set i 1} {$i <= $count} {incr i} {
69         puts "<part lang=\"\$lang($i)\">"
70         foreach {name form} $files($i) {
71             if {[Allowed $form $valid(format)]} {
72                 puts "<file>"
73                 puts "<filename format=\"\$form\">$name</filename>"
74                 if $files(md5) {
75                     if {[catch {set check [lindex [split [exec md5 $name]] 3]} err]} {
76                         puts stderr "Error: cannot generate checksum using md5!"
77                     } else {
78                         puts "<checksum method=\"md5\">$check</checksum>"
79                     }
80                 }
81                 puts "</file>"
82             } else {
83                 puts stderr "Warning: format \"\$form\" not valid, entry skipped!"
84             }
85         }
86         puts "</part>"
87     }
88 }
89
90 # write the info part
91 proc WriteInfo {} {
92     global valid info
93     puts "<info>"
94     puts "<author>$info(author)</author>"
95     puts "<comment>$info(comment)</comment>"
96     puts "<date>$info(date)</date>"

```

```

97     if {[Allowed $info(type) $valid(type)]} {
98         puts "<source type=\"\$info(type)\">$info(source)</source>"
99     } else {
100        puts "<source>$info(source)</source>"
101    }
102    puts "</info>"
103 }
104
105 # check if given argument is valid
106 proc Allowed {format allowed} {
107     foreach v $allowed {
108         if {![string compare $v $format]} {
109             return 1
110         }
111     }
112     return 0
113 }
114
115 # print help message
116 proc Help {} {
117     global valid
118     puts "Usage:\nidx-add.tcl \[-x\] \[info-options\] \[-u <use>\] -p -l <lang> <file1> <format1> <file2> <form
119     puts "General Options:"
120     puts "-x:  short mode, skip document header and footer"
121     puts "-p:  start a new part"
122     puts "-m:  generate id (checksum) using md5"
123     puts "-l:  set language of part"
124     puts "-u:  mark document for use\n"
125     puts "Info Options:"
126     puts "-a:  set author"
127     puts "-c:  set comment"
128     puts "-d:  set date"
129     puts "-s:  set source"
130     puts "-t:  set source type\n"
131     puts "<source_type> may be one of \"\$valid(type)\""
132     puts "<format> may be one of \"\$valid(format)\""
133     puts "<lang> may be one of \"\$valid(lang)\""
134     puts "<use> may be one of \"\$valid(use)\""
135     exit
136 }
137
138 # print version info
139 proc Version {} {
140     global version
141     puts $version
142     exit
143 }
144
145 # -----
146
147 # check if any arguments given
148 if {![llength $argv]} {Help}
149
150 # process command line arguments
151 set flag error
152 foreach arg $argv {
153     switch -exact -- $arg {
154         -a {set flag author}
155         -c {set flag comment}
156         -d {set flag date}
157         -h {Help}

```

```

158 -l {set flag lang}
159 -m {set files(md5) 1}
160 -p {incr count; set flag part}
161 -x {set small 1}
162 -s {set flag source}
163 -t {set flag type}
164 -u {set flag use}
165 -v {Version}
166 default {
167     switch -exact -- $flag {
168         author {set info(author) $arg}
169         comment {set info(comment) $arg}
170         date {set info(date) $arg}
171         error {Help}
172         lang {set lang($count) $arg; set flag part}
173         part {lappend files($count) $arg}
174         source {set info(source) $arg}
175         type {set info(type) $arg}
176         use {set useFlag $arg}
177         default {Help}
178     }
179 }
180 }
181 }
182
183 # generate xml file and print to stdout
184 if $small {SmallHeader} else {Header}
185 WriteParts $count
186 WriteInfo
187 if $small {SmallFooter} else {Footer}

```

I.3.5 idx-repair.tcl

Repariert Rumpf-Datei (s. C.1.3)

```

1 #!/bin/sh
2 # the next line restarts using tclsh \
3 exec tclsh8.3 "$0" "$@"
4
5 # $Header: /opt/cvs/bin/idx-repair.tcl,v 1.1 2001/10/21 15:20:24 koks Exp $
6
7 # repair an index.xml file
8 # KOKS project
9 # Arno Erpenbeck, 11.01.2001
10
11 # load packages
12 lappend auto_path /home/koks/share/tcl
13 package require koksUtil
14
15
16 if $argc {
17     set in [koksUtil::openFile [lindex $argv 0]]
18 } else {
19     set in [koksUtil::openFile "index.xml"]
20 }
21
22 puts "<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>"
23 puts "<!DOCTYPE mapping SYSTEM \"/home/koks/share/mapping.dtd\">"
24 puts "<mapping>"
25 puts [read $in]
26 puts "</mapping>"

```

```

27
28 koksUtil::closeFile $in

```

I.3.6 meta-ls.rb

Listet Meta-Index metaindex.xml (s. C.1.3)

```

1  #!/usr/local/bin/ruby
2
3  require "xmlparser"
4
5  # $Header: /opt/cvs/bin/meta-ls.rb,v 1.2 2001/07/23 14:23:46 koks-cvs Exp $
6
7  # list contents of an metaindex.xml file
8  # KOKS project
9  # Arno Erpenbeck, 23.07.2001
10
11
12 class MyXMLParser < XMLParser
13
14   def initialize(enc)
15     super(enc)
16     @inside = false
17   end
18
19   def character(data)
20     print "\t", data.strip, "\n" if @inside
21   end
22
23   def startElement(name, attrs)
24     if name == "korpus"
25       puts attrs["name"]
26     elsif name == "file"
27       @inside = true
28     end
29   end
30
31   def endElement(name)
32     @inside = false if name == "file"
33   end
34
35 end
36
37 $0 = File.basename($0)
38 if ARGV.length == 0
39   inf = "metaindex.xml"
40 elsif ARGV.length == 1
41   inf = ARGV[0]
42 else
43   STDERR.puts "Usage: #{$0} [meta-index]"
44   exit 1
45 end
46 begin
47   parser = MyXMLParser.new("ISO-8859-1")
48   f = File.open(inf, "r")
49   parser.parse(f.read)
50   f.close
51 rescue
52   STDERR.puts "Hoppla: #{$!}"
53   exit 1
54 end

```


I.4 Sprachklassifikation

I.4.1 ngram.rb

N-Gramme erstellen (s. C.1.4)

```

1  #!/usr/local/bin/ruby
2
3  # $Header: /opt/cvs/artus/ngram.rb,v 1.3 2001/03/28 09:10:52 koks-cvs Exp $
4
5  # ngram.rb - build up n-gram table for text file
6  #
7  # KOKS project
8  # Arno Erpenbeck, 01.03.2001
9
10 require 'optparse'
11 require 'math/statistics'
12
13 # count occurrences of n-grams
14 class Hash
15   include Math::Statistics
16   Hash::default_block = lambda{|i,j| j}
17   def inc(key)
18     prev = fetch(key, 0)
19     store(key, prev+1)
20   end
21 end
22
23 # make n-grams of arbitrary size of a given string
24 class String
25
26   def normalize
27     self.gsub(/[äöüÄÖÜß]/,'#').gsub(/\s/,' ').downcase
28   end
29
30   def ngrams(len=1)
31     ngrams = []
32     (0..size - len).each do |n|
33       ng = self[n..n+len]
34       if ng =~ /[a-zA-ZäöüßA-ZÄÖÜ\s]{3,3}/
35         ngrams.push(ng)
36         yield ng if block_given?
37       end
38     end
39     ngrams
40   end
41 end
42
43 # read file and build up n-grams
44 class Shredder
45
46   def initialize(infile, outfile, bytes=-1, normalize=false)
47     @infile = infile
48     @outfile = outfile
49     @bytes = bytes
50     @normalize = normalize
51   end
52
53   def run(len=3)
54     count = 0
55     hash = {}

```

```

57     if @normalize
58         while line = @infile.gets
59             line.ngrams(len) do |s| hash.inc(s.normalize) end
60             if @bytes > 0
61                 count += line.length
62                 break if count >= @bytes
63             end
64         end
65     else
66         while line = @infile.gets
67             line.ngrams(len) do |s| hash.inc(s) end
68             if @bytes > 0
69                 count += line.length
70                 break if count >= @bytes
71             end
72         end
73     end
74     s = Float.induced_from(hash.size)
75     #hash.sort.each {|k,v| @outfile.print k,":",v/s,"\n"}
76     hash.sort.each {|k,v| @outfile.print k,":",v,"\n"}
77     #@outfile.print hash.avg,"\n"
78 end
79
80 end
81
82 # check command line args
83 $0 = File.basename($0)
84 vars = {:len => 3, :bytes => -1, :norm => false}
85 parser = OptionParser.new do |q|
86     q.banner = "Usage: #{$0} [options]\n"
87     q.on_tail("-h", "--help", "show this message") {puts q; exit}
88     q.on("-i", "--input=FILENAME", String,
89         "input file name") {|vars[:infile]|}
90     q.on("-o", "--output=FILENAME", String,
91         "output file name") {|vars[:outfile]|}
92     q.on("-b", "--bytes=VALUE", Integer,
93         "read at most VALUE bytes") {|vars[:bytes]|}
94     q.on("-l", "--len=VALUE", Integer,
95         "build n-grams of VALUE length") {|vars[:len]|}
96     q.on("-n", "--normalize",
97         "normalize n-grams") {|vars[:norm]|}
98 end
99 begin
100     parser.parse!(ARGV)
101 rescue OptionParser::ParseError
102     STDERR.puts parser
103     exit 1
104 end
105
106 begin
107     # open input and output files
108     if vars[:infile]
109         infile = open(vars[:infile], "r")
110     else
111         infile = STDIN
112     end
113     if vars[:outfile]
114         outfile = open(vars[:outfile], "w+")
115     else
116         outfile = STDOUT
117     end

```

```

118   Shredder.new(infile, outfile, vars[:bytes], vars[:norm]).run(vars[:len])
119 rescue
120   STDERR.puts "Something bad happened: #{!}\n"
121 ensure
122   # close file if necessary
123   infile.close if vars[:infile]
124   outfile.close if vars[:outfile]
125 end
126

```

I.4.2 classifier.rb

Klassifiziert Daten (s. C.1.4)

```

1  # $Header: /opt/cvs/artus/classifier.rb,v 1.2 2001/03/12 14:13:34 koks-cvs Exp $
2
3  # classifier.rb - try to identify the language of a text
4  #
5  # KOKS project
6  # Arno Erpenbeck, 11.03.2001
7
8  # find index of smallest/biggest value in array
9  class Array
10   def max(start = 0)
11     idx = start
12     max = self[idx]
13     self.each_index do |i|
14       if self[i] > max
15         max = self[i]
16         idx = i
17       end
18     end
19     idx
20   end
21
22   def min(start = 0)
23     idx = start
24     min = self[idx]
25     self.each_index do |i|
26       if self[i] < min
27         min = self[i]
28         idx = i
29       end
30     end
31     idx
32   end
33 end
34
35 # read file and build up n-grams
36 class Classifier
37
38   def initialize(datafile, verbose=false)
39     @datafile = datafile
40     @verbose = verbose
41     @q = []
42     @sum = []
43     @id = []
44   end
45
46   # read training data files
47

```

```

48 def read_data
49   # read file containing ids and names of training data files
50   fl = []
51   begin
52     dfin = open(@datafile, "r")
53     dfin.read.scan(/([a-z]+\s([\w.\-]+))/) do |id, fnm|
54       fl << fnm
55       @id << id
56     end
57   rescue
58     STDERR.puts "Something bad happened: #{$!}\n"
59   rescue
60     dfin.close
61   end
62   # open each file containg n-grams of training data
63   fl.each_index do |idx|
64     print "Reading data file '", fl[idx], "'..." if @verbose
65     @q[idx] = {}
66     @sum[idx] = 0
67     begin
68       infile = open(fl[idx], "r")
69       infile.read.scan(/([a-z# ]+):(\d+)/) do |k, v|
70         @q[idx].store(k, v)
71         @sum[idx] += v.to_i
72       end
73     rescue
74       STDERR.puts "Something bad happened: #{$!}\n"
75     ensure
76       infile.close if infile
77     end
78     print "(#{@q[idx].size}, #{@sum[idx]})\n" if @verbose
79   end
80 end
81
82 # smooth training data
83 def smooth_data
84   print "Smoothing training data...\n" if @verbose
85   @q.each_index do |idx1|
86     @q[idx1].each_key do |k|
87       @q.each_index do |idx2|
88         if idx1 != idx2 and !@q[idx2].has_key?(k)
89           @q[idx2].store(k, 0.5)
90           @sum[idx2] += 0.5
91         end
92       end
93     end
94   end
95   print "Normalizing training data...\n" if @verbose
96   @q.each_index do |idx|
97     div = 1.0 / @sum[idx]
98     @q[idx].each do |k, v|
99       @q[idx].store(k, v.to_f * div)
100    end
101  end
102 end
103
104 # classify test data from hash
105 def classify(p)
106
107   # clean test data
108   print "Cleaning test data...\n" if @verbose

```

```

109 p.each_key do |k|
110   delete = TRUE
111   @q.each_index do |i|
112     if @q[i].has_key?(k)
113       delete = FALSE
114       break
115     end
116   end
117   if delete
118     p.delete(k)
119   end
120 end
121
122 # normalize data
123 print "Normalizing test data...\n" if @verbose
124 sum = 0
125 p.each do |k,v|
126   sum += v.to_i
127 end
128 div = 1.0 / sum
129 p.each do |k,v|
130   p.store(k, v.to_f * div)
131 end
132
133 # calculate Kullback-Leibler distance
134 print "Calculating distances...\n" if @verbose
135 dist = Array.new(@q.size)
136 @q.each_index do |i|
137   dist[i] = 0.0
138   p.each do |k, v|
139     w = v.to_f
140     #print "Ohje: ",k,"\n" if !@q[i].has_key?(k)
141     z = @q[i].fetch(k).to_f
142     dist[i] = dist[i] + (w * Math.log(w/z))
143   end
144   print "dist[\",i,\"] = \",dist[i],\"\n" if @verbose
145 end
146
147 # find minimum -> return language classification
148 @id[dist.min(1)]
149
150 end
151
152 end

```

I.4.3 cfile.rb

Sprache einer Textdatei ermitteln (s. C.1.4)

```

1 #!/usr/local/bin/ruby
2
3 # $Header: /opt/cvs/artus/cfile.rb,v 1.2 2001/03/28 09:10:52 koks-cvs Exp $
4
5 # cfile.rb - try to identify the language of a text (file version)
6 #
7 # KOKS project
8 # Arno Erpenbeck, 11.03.2001
9
10 require 'optparse'
11 require 'classifier'
12

```

```

13 class FileReader
14
15   def initialize(testfile, classifier, verbose=false)
16     @testfile = testfile
17     @classifier = classifier
18     @verbose = verbose
19   end
20
21   # read test data files and classify each file
22   def classify_input
23     fl = []
24     # read file containing names of test data files
25     begin
26       dfin = open(@testfile, "r")
27       while dfin.gets
28         fl << $_.chomp!
29       end
30     rescue
31       STDERR.puts "Something bad happened: #{$!}\n"
32     rescue
33       dfin.close
34     end
35
36     p = {}
37     fl.each do |fnm|
38       # open each test data file and classify
39       print "Reading test file '", fnm, "'...\n" if @verbose
40       p.clear
41       begin
42         infile = open(fnm, "r")
43         infile.read.scan(/([a-z# ]+):(\d+)/) {|k, v| p.store(k, v)}
44       rescue
45         STDERR.puts "Something bad happened: #{$!}\n"
46       ensure
47         infile.close if infile
48       end
49       # call classifier
50       id = @classifier.classify(p)
51       print fnm,": ",id,"\n"
52       yield id if block_given?
53     end
54   end
55
56 end
57
58 # check command line args
59 $0 = File.basename($0)
60 vars = {:datafile => "datafiles", :testfile => "testfiles",
61         :verbose => false, :smooth => true}
62 parser = OptionParser.new do |q|
63   q.banner = "Usage: #{$0} [options]\n"
64   q.on_tail("-h", "--help", "show this message") {puts q; exit}
65   q.on("-d", "--data=FILENAME", String,
66        "data file names [default: 'datafiles']") {|vars[:datafile]|}
67   q.on("-t", "--test=FILENAME", String,
68        "test file name [default: 'testfiles']") {|vars[:testfile]|}
69   q.on("-n", "--nosmooth", "do not smooth training data") {|vars[:nosmooth]|}
70   q.on("-v", "--verbose", "be verbose") {|vars[:verbose]|}
71 end
72 begin
73   parser.parse!(ARGV)

```

```

74 rescue OptionParser::ParseError
75   STDERR.puts parser
76   exit 1
77 end
78
79 begin
80   c = Classifier.new(vars[:datafile], vars[:verbose])
81   c.read_data
82   c.smooth_data unless vars[:nosmooth]
83   f = FileReader.new(vars[:testfile], c, vars[:verbose])
84   f.classify_input
85   #f.classify_input {|res| print "==> ",res,"\n"}
86 rescue
87   STDERR.puts "Something bad happened: #{$!}\n"
88 ensure
89 end
90

```

I.4.4 cserver.rb

Klassifikationsserver (s. C.1.4)

```

1  #!/usr/local/bin/ruby
2
3  # $Header: /opt/cvs/artus/cserver.rb,v 1.2 2001/03/23 13:10:03 koks-cvs Exp $
4
5  # cserver.rb - try to identify the language of a text (server version)
6  #
7  # KOKS project
8  # Arno Erpenbeck, 11.03.2001
9
10 require 'optparse'
11 require 'socket'
12 require 'classifier'
13
14 class SocketReader
15
16   def initialize(socket, classifier, verbose=true)
17     @socket = socket
18     @classifier = classifier
19     @verbose = verbose
20   end
21
22   # read test data from socket and classify
23   def classify_input
24     print "Reading test data from ",@socket,"...\n" if @verbose
25     p = {}
26     begin
27       while @socket.gets
28         if $_ =~ "END" ; break ; end
29         $_.scan(/([a-z# ]+):(\d+)/) {|k, v| p.store(k, v)}
30       end
31       print "Read ",p.size," data pairs\n" if @verbose
32     rescue
33       STDERR.puts "Something bad happened: #{$!}\n"
34     ensure
35       #@socket.close
36     end
37     @classifier.classify(p)
38   end
39

```

```

40 end
41
42 # check command line args
43 $0 = File.basename($0)
44 vars = {:datafile => "datafiles", :verbose => false, :nosmooth => false,
45   :port => 9999}
46 parser = OptionParser.new do |q|
47   q.banner = "Usage: #{$0} [options]\n"
48   q.on_tail("-h", "--help", "show this message") {puts q; exit}
49   q.on("-d", "--data=FILENAME", String,
50     "data file names") {|vars[:datafile]|}
51   q.on("-p", "--port=PORT", Integer,
52     "listening port") {|vars[:port]|}
53   q.on("-n", "--nosmooth",
54     "do not smooth training data") {|vars[:nosmooth]|}
55   q.on("-v", "--verbose",
56     "be verbose") {|vars[:verbose]|}
57 end
58 begin
59   parser.parse!(ARGV)
60 rescue OptionParser::ParseError
61   STDERR.puts parser
62   exit 1
63 end
64
65 begin
66   c = Classifier.new(vars[:datafile], vars[:verbose])
67   c.read_data
68   c.smooth_data unless vars[:nosmooth]
69   # open server socket
70   gs = TCPServer.open("nunix.cl-ki.uni-osnabrueck.de", vars[:port])
71   addr = gs.addr
72   addr.shift
73   print Time.now, ": cserver listening on ", addr.join(":"), "... \n"
74   # wait for incoming connections
75   while true do
76     ns = gs.accept
77     print Time.now, ": accepting connecting ", ns, "... \n"
78     # start new thread for each connection
79     Thread.start do
80       socket = ns
81       reader = SocketReader.new(socket, c)
82       id = reader.classify_input
83       # write out result
84       socket.write(id)
85       socket.close
86       STDOUT.flush
87     end
88   end
89 rescue
90   STDERR.puts "Something bad happened: #{$!}\n"
91 ensure
92   gs.close
93 end

```

I.4.5 cclient.rb

Klassifikationsclient (s. C.1.4)

```

1 #!/usr/local/bin/ruby
2

```



```
3 # $Header: /opt/cvs/artus/cclient.rb,v 1.1 2001/03/12 14:14:12 koks-cvs Exp $
4
5 # cclient.rb - try to identify the language of a text (client version)
6 #
7 # KOKS project
8 # Arno Erpenbeck, 11.03.2001
9
10 require 'optparse'
11 require 'socket'
12
13 # check command line args
14 $0 = File.basename($0)
15 vars = {:testfile => "testfile", :server => "localhost", :port => 9999}
16 parser = OptionParser.new do |q|
17   q.banner = "Usage: #{$0} [options]\n"
18   q.on_tail("-h", "--help", "show this message") {puts q; exit}
19   q.on("-t", "--test=FILENAME", String,
20     "test file names [default: 'testfile']") {|vars[:testfile]|}
21   q.on("-s", "--server=ADDRESS", String,
22     "server address") {|vars[:server]|}
23   q.on("-p", "--port=PORT", Integer,
24     "server port") {|vars[:port]|}
25 end
26 begin
27   parser.parse!(ARGV)
28 rescue OptionParser::ParseError
29   STDERR.puts parser
30   exit 1
31 end
32
33 begin
34   # open socket to specified server
35   socket = TCPSocket.open(vars[:server], vars[:port])
36   # read in test file data and write to socket
37   infile = open(vars[:testfile], "r")
38   while infile.gets
39     socket.write($_)
40   end
41   # signal end of data
42   socket.write("END\n")
43   socket.flush
44   # wait for result
45   id = socket.gets
46   print "==> ",id,"\n"
47 rescue
48   STDERR.puts "Something bad happened: #{$!}\n"
49 ensure
50   socket.close
51 end
```

Anhang J

Wörterbuch-Code

J.1 Code für die Dictionary-Entry-Parser

J.1.1 Beispielcode für einfache Wörterbücher: vsource.bas

Dieses Programm steht stellvertretend für alle Dictionary Entry Parser, welche Wörterbücher des Einfachen Fall"bearbeiteten.

```
1 Private Sub DateienBearbeiten()  
2 'Norman Kummer 01-03 2001  
3 'Bsp. für einen [D]ictionary[E]ntry[P]arser für Formate wie unten beschrieben.  
4  
5 'org.:(input)  
6 'dt:: eng  
7 'speziell abk.:  
8 ' dt abk.:dt voll::ebg abk.:eng voll  
9 'neu:(output)  
10 'dt::eng##rest  
11 '  
12 'rest:  
13 ' {dt};;{eng};;(dt);;(eng);;[dt];;[eng];;<dt>;<eng>  
14 'aFormat:  
15 ' {d}info;;{e}info;;(d)info;;(e)info;;[d]info;;[e]info;;<d>info;;<e>info  
16 '  
17 'anmerkung:  
18 '*manchmal in eng (;) verschiedene weitere Möglichkeiten z.B.:  
19 ' jemanden davon abhalten; etwas zu tun :: to keep (prevent; stop) someone from doing something  
20 ' im dt aber (;) erläuterungen!!!  
21 '*möglichlicherweise letztes wort ersetzbar bei () -> also neue zeile  
22 '*... lass ich weg, manchmal....  
23 '*bei verben / -> kombinationen bilden  
24 '*nachbearbeiten bei [alt], sonst vielleicht schwierigkeiten mit korpora  
25 ' dafür neue einträge bilden, oder speziell abfragen beim parsen  
26 '*kombinationen mit , z.B.:  
27 ' - : Pfund (Gewicht) :: lb, lbs : pound, pounds  
28 '* - bedeutet: keine info, oft' - ' auch ' ? '  
29 '* fehler:  
30 ' - Bin ich Jesus? :: AIJ : Bin ich Jesus?  
31  
32  
33 'schritte:  
34 '* suchen nach '::'  
35 ' - teilen in dt./eng.  
36 ' - nur behandeln, wenn beide seiten mehr als ' '  
37 ' - falls noch ':' vorhanden -> sonderbehandlung abk.  
38 '* suchen auf beiden seiten nach ';' '  
39 ' - neue zeilen draus machen + voller rest
```

```

40 ' - auf seiten alle klammern rausschmeißen
41 Dim Datei As String
42 Dim Textzeile As String
43 Dim Ausgabezeile As String
44 Dim Dt As String
45 Dim Eng As String
46 Dim DtEngFeld(1, 10) As String
47 Dim RestAusgabe As String
48
49 Dim SuchZeichen As String
50 Dim Pos As Integer
51 Dim AlterName As String
52 Dim NeuerName As String
53 Dim i As Integer
54 Dim j As Integer
55 Dim Pfad As String
56 Pfad = "..\"
57 Datei = "ger-eng.txt"
58
59 'öffnet Eingabedateien
60 Open Pfad & Datei For Input As #1 ' Datei öffnen.
61 FileLen (Pfad & Datei)
62 ProgressBar.Min = 0
63 ProgressBar.Value = ProgressBar.Min
64 ProgressBar.Max = FileLen(Pfad & Datei)
65 'öffnet Ausgabedateien
66 Open Pfad & "KOKSengdt_" & Datei For Output As #2
67 Open Pfad & "KOKSdt_" & Datei For Output As #3
68 Open Pfad & "KOKSeng_" & Datei For Output As #4
69 Do While Not EOF(1) ' Schleife bis Dateiende.
70 Line Input #1, Textzeile ' Zeile in Variable einlesen.
71 ProgressBar.Value = ProgressBar.Value + Len(Textzeile)
72 '* suchen nach ':' Delimiter1
73 If Left(Textzeile, 1) <> "#" Then 'rem's ausschließen
74 SuchZeichen = " :: "
75 'Debug.Print Textzeile
76 Pos = InStr(Textzeile, SuchZeichen)
77 If Pos > 0 Then
78 ' - teilen in dt./eng.
79 ' - auf beiden seiten alle klammern rausschmeißen
80 Dt = KeineKlammern(Left(Textzeile, Pos - 1))
81 Eng = KeineKlammern(Right(Textzeile, Len(Textzeile) - Pos + 1 - Len(SuchZeichen)))
82 ' - nur behandeln wenn beide seiten mehr als ' '
83 If Dt <> " " And Eng <> " " Then
84 RestAusgabe = Textzeile
85 ' - falls noch ':' vorhanden -> sonderbehandlung abk.(bei Wörterbuch 'Ding' vorhanden)
86 SuchZeichen = ":"
87 Pos = InStr(Dt, SuchZeichen) 'auch wenn eng. abk.
88 If Pos > 0 Then
89 'abk.
90 Else
91 '* suchen auf beiden seiten nach ';' Delimiter2
92 Dim Seite(1) As String
93 Dim LetztePos As Integer
94 Dim Zähler As Integer
95 Dim Spezl As Integer
96 LetztePos = 1
97 Seite(0) = Dt 'bei i=0 für dt
98 Seite(1) = Eng 'bei i=1 für eng
99 Pos = 1
100 For i = 0 To 1

```

```

101         Zähler = 0
102         SuchZeichen = ";"
103         Spezl = -1
104         Do
105     ' - neue zeilen draus machen + voller rest
106         'dt/eng felder aufbauen
107
108         SuchZeichen = ";"
109         Pos = InStr(LetztePos, Seite(i), SuchZeichen)
110         If Pos = 0 Then
111             Pos = Len(Seite(i))
112             Spezl = -1
113         Else
114             Spezl = 0
115         End If
116         DtEngFeld(i, Zähler) = Mid(Seite(i), LetztePos, Pos - LetztePos - Spezl)
117         If Right(DtEngFeld(i, Zähler), 1) = " " Then
118             DtEngFeld(i, Zähler) = Left(DtEngFeld(i, Zähler), Len(DtEngFeld(i, Zähler)
119         End If
120         Zähler = Zähler + 1
121         LetztePos = Pos + 2
122         Spezl = -1
123     Loop Until Pos = Len(Seite(i))
124     Pos = 1
125     LetztePos = 1
126     'vorber
127 Next
128 'dt eng kombinieren und ausgeben
129 'neu:
130 'dt::eng##rest
131 For i = 0 To 10
132     If DtEngFeld(0, i) <> "" Then
133         For j = 0 To 10
134             If DtEngFeld(1, j) <> "" Then
135                 If Right(DtEngFeld(0, i), 1) = " " Then
136                     DtEngFeld(0, i) = Left(DtEngFeld(0, i), Len(DtEngFeld(0, i)) - 1)
137                 End If
138                 If Right(DtEngFeld(1, j), 1) = " " Then
139                     DtEngFeld(1, j) = Left(DtEngFeld(1, j), Len(DtEngFeld(1, j)) - 1)
140                 End If
141                 Print #3, DtEngFeld(0, i) & "<KOKS>. <ABSATZ></KOKS>"
142                 Print #4, DtEngFeld(1, j) & "<KOKS>. <ABSATZ></KOKS>"
143                 Textzeile = DtEngFeld(0, i) & "::" & DtEngFeld(1, j) & "##" & RestAus
144                 If Left(RestAusgabe, 1) <> Left(Textzeile, 1) Then
145                     'Debug.Print Textzeile
146                     DoEvents
147                 End If
148                 Print #2, Textzeile
149             Else
150                 j = 10
151             End If
152         Next j
153     Else
154         i = 10
155     End If
156 Next i
157 For i = 0 To 10
158     For j = 0 To 10
159         DtEngFeld(0, i) = ""
160         DtEngFeld(1, j) = ""
161     Next

```

```

162         Next
163     End If
164 End If
165 End If
166 End If
167 Loop
168 'Dateien schließen
169 Close #1
170 Close #2
171 Close #3
172 Close #4
173 MsgBox "Fertig!"
174 End Sub
175
176 Function KeineKlammern(Textzeile As String) As String
177 'löscht alle Klammern aus dem String 'Textzeile'
178 Dim Klammern(3) As String
179 Dim Pos As Integer
180 Dim Pos2 As Integer
181 Dim SuchZeichen As String
182 Dim i As Integer
183 Dim Textzeile2 As String
184 Textzeile2 = Textzeile
185 Klammern(0) = "("
186 Klammern(1) = "["
187 Klammern(2) = "{"
188 Klammern(3) = "<>"
189 For i = 0 To 3
190     Pos = 1
191     Do While Pos > 0
192         SuchZeichen = Left(Klammern(i), 1)
193         Pos = InStr(Textzeile2, SuchZeichen)
194         SuchZeichen = Right(Klammern(i), 1)
195         Pos2 = InStr(Textzeile2, SuchZeichen)
196
197         If Pos > 0 Then
198             Textzeile2 = Left(Textzeile2, Pos - 1) & Right(Textzeile2, Len(Textzeile2) - Pos2)
199         End If
200     Loop
201 Next i
202 If Right(Textzeile2, 1) = " " Then
203     Textzeile2 = Left(Textzeile2, Len(Textzeile2) - 1)
204 End If
205 KeineKlammern = Textzeile2
206 End Function
207
208 Private Sub Command1_Click()
209     DateienBearbeiten
210 End Sub

```

J.1.2 Code für das LQL-Wörterbuch

Diese Scripte wurden immer wieder angepaßt, bis das Lexikon die zum Einlesen benötigte Form hatte. Auch wurden vorhandene Fehler zusätzlich mit dem VI (Ersetzen/Löschen) ausgemerzt.

Schlüssel/werte aus Rohdaten aufsammeln: collect1.rb, collect2.rb, collect3.rb

```

1  #!/usr/local/bin/ruby
2
3  if ARGV.size == 1
4      infile = open(ARGV[0], "r")
5  else
6      infile = STDIN
7  end
8
9  keys = []
10
11 while line = infile.gets
12     if line =~ /^(^ |\()([a-z]+):/
13         #puts $1
14         keys << $2 unless keys.include?($2)
15     end
16 end
17
18 infile.close if infile != STDIN
19
20 keys = keys.sort
21 keys.each {|idx| puts idx }
22 puts keys.size

```

```

1  #!/usr/local/bin/ruby
2
3  if ARGV.size == 1
4      infile = open(ARGV[0], "r")
5  else
6      infile = STDIN
7  end
8
9  keys = []
10
11 while line = infile.gets
12     line.scan(/( |:)([A-ZÄÖÛßäöüßa-z_-]+):/) { |match|
13         #line.scan(/([A-ZÄÖÛßäöüßa-z_]+)\(/) { |match|
14             keys << match[1] unless keys.include?(match[1])
15         }
16 end
17
18 infile.close if infile != STDIN
19
20 keys = keys.sort
21 keys.each {|idx| puts idx }
22 puts keys.size

```

```

1  #!/usr/local/bin/ruby
2
3  if ARGV.size == 1
4      infile = open(ARGV[0], "r")
5  else

```

```
6     infile = STDIN
7 end
8
9 keys = []
10
11 while line = infile.gets
12     if line =~ /usage_note:([\w ]+)/
13         puts $1
14     end
15     #line.scan(/usage_note:([A-Za-z ]+\()/) { |match|
16         #print match
17         #keys << match unless keys.include?(match)
18     #}
19 end
20
21 infile.close if infile != STDIN
22
23 skeys = keys.sort
24 skeys.each {|idx| puts idx }
25 puts skeys.size
```

”collocat” Phrasen aus Lexikon aufsammeln: collocat.rb

```
1  #!/usr/local/bin/ruby
2
3  =begin
4  collocat.rb
5  'collocat'-Phrasen aufsammeln
6  Norman Kummer, Arno Erpenbeck
7  KOKS Projekt, April/Mai 01
8  =end
9
10 require 'string-utils'
11
12 if ARGV.size == 1
13     infile = open(ARGV[0], "r")
14 else
15     infile = STDIN
16 end
17
18 class Phrase
19     attr_accessor :key
20
21     def initialize(key=nil)
22         key = "" unless key
23         @key = key
24         @source = []
25         @target = []
26         @current = 0
27     end
28
29     def reset_all
30         @source = []
31         @target = []
32     end
33
34     def reset
35         @target[@current] = []
36     end
37
38     def source=(what)
39         @source << what
40     end
41
42     def target=(what)
43         @target[@current] << what
44     end
45
46     def fix_target
47         @target[@current].last << ")" unless @target[@current].last[-1,1] == ')'
48     end
49
50     def proceed
51         @current += 1
52     end
53
54     def dump
55         print @key, "\t"
56         @source.each {|x| print x, " "}
57         print "\t"
58         @target[@current].each {|x| print x, " "}
59         print "\n"
```



```

60     end
61
62 end
63
64 lines = 0
65
66 # jede Zeile lesen und auseinander nehmen
67 while line = infile.gets
68     lines += 1
69     STDERR.print "." if lines % 1000 == 0
70     modus = :egal # Modus
71     found = false # Deutsches Wort gefunden
72     count = 0 # Klammerzaehler
73     saved = 0
74     extra = 0
75     phrase = Phrase.new
76     # alle Tokens in einer Zeile anschauen
77     line.split.each { |token|
78         count += token.count_braces
79         # erstes Token ist deutsch
80         if !found
81             phrase.key = token
82             found = true
83         # wenn 'collocat(' kommt wird es spannend
84         elsif token == "collocat("
85             modus = :coll
86         # 'source:' gefunden -> weiterlaufen und aufsammeln
87         elsif modus == :coll && token == "source:"
88             modus = :source
89             phrase.reset_all
90         # aufsammeln bis zum key:
91         elsif modus == :source
92             if token.is_key?
93                 modus = :skip
94             elsif token == "targ("
95                 modus = :targ
96             else
97                 phrase.source = token
98             end
99         elsif modus == :skip && token == "targ("
100             modus = :targ
101         elsif modus == :targ && token == "target("
102             modus = :target
103             saved = count
104         elsif modus == :target
105             if token == "word:" || token == "phrase:"
106                 modus = :word
107                 extra = 0
108                 phrase.reset
109             end
110         elsif modus == :word
111             # zu weit gelaufen -> modus zurück setzen
112             if token.is_key? || count < saved
113                 phrase.target = token.clear(extra) if token.is_end?
114                 phrase.fix_target if extra > 0
115                 phrase.dump
116                 phrase.proceed
117                 modus = :skip
118             else
119                 # Klammergebirge nachher reparieren
120                 if token.is_start? && !token.is_end?

```

```
121             extra = 1
122         elsif token.is_end?
123             extra = 0
124         end
125         phrase.target = token
126     end
127 end
128 }
129 end
130
131 infile.close if infile != STDIN
```

'~' in Phrasen durch Schlüsselwort ersetzen: ersetzer.rb

```

1 #!/usr/local/bin/ruby
2
3 =begin
4   ersetzer.rb
5   '~' in Phrasen durch Wort ersetzen
6   Norman Kummer, Arno Erpenbeck
7   KOKS Projekt, April/Mai 01
8 =end
9
10 class String
11
12     def replace(str)
13         pos = self.index('~')
14         if !pos
15             # kein '~' in Wort
16             return self
17         elsif pos == 0
18             # '~' am Wortanfang -> ersetzen
19             self.gsub(/~/, str)
20         else
21             # '~' nicht am Wortanfang -> kopieren und Teil ersetzen
22             pre = self[0, pos]
23             post = self[pos, self.length].gsub(/~/, str[pos, str.length])
24             pre << post
25         end
26     end
27
28 end
29
30 # open input and output files
31 if ARGV.length != 3
32     STDERR.puts "Usage: ersetzer <lex-file> <german-out> <english-out>"
33     exit 1
34 end
35 if ARGV[0] == "-"
36     infile = STDIN
37 else
38     infile = open(ARGV[0], "r")
39 end
40 gfile = open(ARGV[1], "w")
41 efile = open(ARGV[2], "w")
42
43 # Zeilen verarbeiten, dabei '~' durch keyword ersetzen
44 lines = 0
45 while line = infile.gets
46     lines += 1
47     tokens = line.split(/\t/)
48     if tokens.size != 3
49         STDERR.print("Illegal line: #{lines}\n")
50         #exit 2
51         next
52     end
53     german = tokens[0]
54     tokens[1].split.each {|token| gfile.print token.strip.replace(german), " " }
55     gfile.print "\n"
56     tokens[2].each {|token| efile.print token.strip, " " }
57     efile.print "\n"
58 end
59

```

```
60 # aufräumen
61 if infile != STDIN
62     infile.close
63 end
64 gfile.close
65 efile.close
```

Übersetzungspaare aufsammeln (eng-ger): etran.rb

```

1 #!/usr/local/bin/ruby
2
3 =begin
4   etran.rb
5   'tran'-Phrasen aufsammeln (eng-ger)
6   Norman Kummer, Arno Erpenbeck
7   KOKS Projekt, April/Mai 2001
8 =end
9
10 require 'string-utils'
11 require 'translation'
12
13 if ARGV.size == 1
14     infile = open(ARGV[0], "r")
15 else
16     infile = STDIN
17 end
18
19 lines = 0
20 # jede Zeile lesen und auseinander nehmen
21 while line = infile.gets
22     lines += 1
23     STDERR.print "." if lines % 1000 == 0
24     modus = :egal # Modus
25     count = 0 # Klammerzaehler
26     extra = 0
27     saved = 0
28     trans = Translation.new
29     # alle Tokens in einer Zeile anschauen
30     line.split.each {|token|
31         count += token.count_braces
32         # solange englisch, bis '^('
33         if token == '^('
34             modus = :skip
35         elsif modus == :egal
36             trans.orig = token
37             # wenn 'hom(' kommt auf 'pos:' warten
38             elsif token == "hom("
39                 modus = :hom
40                 trans.reset_pos
41             # wenn 'tran(' kommt wird es spannend
42             elsif token == "tran("
43                 modus = :tran
44                 saved = count
45             # 'pos:' gefunden -> Wortart aufsammeln
46             elsif modus == :hom && token == "pos:"
47                 modus = :pos
48             # ersten Wert nach 'pos:' behalten
49             elsif modus == :pos
50                 if token.is_key? || token.is_brace?
51                     modus = :skip
52                 else
53                     trans.pos = token
54                 end
55             # 'word:' gefunden -> weiterlaufen und aufsammeln
56             elsif modus == :tran && token == "word:"
57                 modus = :word
58                 extra = 0
59                 trans.reset_tran

```

```
60         # 'phrase:' gefunden -> weiterlaufen und aufsammeln
61     elsif modus == :tran && token == "phrase:"
62         modus = :phrase
63     # aufsammeln bis zum Ende
64     elsif modus == :word
65         # zu weit gelaufen -> modus zurück setzen
66         if token.is_key? || token.is_brace? || count < saved
67             trans.tran = token.clear(extra) if token.is_end?
68             trans.fix_tran if extra > 0
69             trans.dump
70             modus = :skip
71         else
72             # Klammergebirge nachher reparieren
73             if token.is_start? && token.count_braces > 0 && !token.is_end?
74                 extra = 1
75             elsif token.is_end?
76                 extra = 0
77             end
78             trans.tran = token
79         end
80     end
81 }
82 end
83
84 infile.close if infile != STDIN
85
```

Übersetzungspaare aufsammeln (ger-eng): gtran.rb

```

1  #!/usr/local/bin/ruby
2
3  =begin
4  gtran.rb
5  'tran'-Phrasen aufsammeln (ger-eng)
6  Norman Kummer, Arno Erpenbeck
7  KOKS Projekt, April/Mai 2001
8  =end
9
10 require 'string-utils'
11 require 'translation'
12
13 if ARGV.size == 1
14     infile = open(ARGV[0], "r")
15 else
16     infile = STDIN
17 end
18
19 lines = 0
20
21 # jede Zeile lesen und auseinander nehmen
22 while line = infile.gets
23     lines += 1
24     STDERR.print "." if lines % 1000 == 0
25     modus = :egal # Modus
26     found = false # Deutsches Wort gefunden
27     count = 0 # Klammerzaehler
28     extra = 0
29     saved = 0
30     trans = Translation.new
31     # alle Tokens in einer Zeile anschauen
32     line.split.each {|token|
33         count += token.count_braces
34         # erstes Token ist deutsch
35         if !found
36             trans.orig = token
37             found = true
38         # wenn 'hom(' kommt auf 'pos:' warten
39         elsif token == "hom("
40             modus = :hom
41             trans.reset_pos
42         # wenn 'tran(' kommt wird es spannend
43         elsif token == "tran("
44             modus = :tran
45             saved = count
46         # 'pos:' gefunden -> Wortart aufsammeln
47         elsif modus == :hom && token == "pos:"
48             modus = :pos
49         # ersten Wert nach 'pos:' behalten
50         elsif modus == :pos
51             if token.is_key? || token.is_brace?
52                 modus = :egal
53             else
54                 trans.pos = token
55             end
56         # 'word:' gefunden -> weiterlaufen und aufsammeln
57         elsif modus == :tran && token == "word:"
58             modus = :word
59             extra = 0

```

```
60         trans.reset_tran
61     # 'phrase:' gefunden -> weiterlaufen und aufsammeln
62     elsif modus == :tran && token == "phrase:"
63         modus = :phrase
64     # aufsammeln bis zum Ende
65     elsif modus == :word
66         # zu weit gelaufen -> modus zurück setzen
67         if token.is_key? || token.is_brace? || count < saved
68             trans.tran = token.clear(extra) if token.is_end?
69             trans.fix_tran if extra > 0
70             trans.dump
71             modus = :egal
72         else
73             # Klammergebirge nachher reparieren
74             if token.is_start? && !token.is_end?
75                 extra = 1
76             elsif token.is_end?
77                 extra = 0
78             end
79             trans.tran = token
80         end
81     end
82 }
83 end
84
85 infile.close if infile != STDIN
```


Tags von LQL auf IMS abbilden: mapper.rb

Das LQL-Wörterbuch hatte eigene Taginformationen, welche mit diesem Script auf das IMS-Set gemappt wurden.

```
1 #!/usr/local/bin/ruby
2
3 =begin
4   mapper.rb
5   Tags von LQL auf eigenes Tagset abbilden
6   Arno Erpenbeck, Norman Kummer
7   KOKS Projekt, April/Mai 2001
8 =end
9
10 # ? noch diskutieren
11 # ! von Hand ändern
12 # " löschen
13
14 hash = {
15   "" => "unknown",
16   "adj" => "adj",
17   "adj attr usu superl" => "adj",
18   "adj usu attr" => "adj",
19   "adj usu pred" => "adj",
20   "addj pred" => "adj",
21   "adj pred, adv" => "adj",
22   "adj pred, adv lines, ropes" => "adj",
23   "adj suf" => "adj",
24   "adj usu attr happiness" => "adj",
25   "adj, interj" => "adj",
26   "adv" => "adv",
27   "adv see adj" => "adv",
28   "adv suf" => "adv",
29   "adv, interj" => "adv",
30   "conj" => "kon",
31   "defective vb" => "unknown",
32   "f or" => "nn",
33   "imper" => "imp",
34   "in" => "unknown",
35   "in cpds" => "unknown",
36   "in cpds mit adj" => "unknown",
37   "in cpds mit superl" => "unknown",
38   "in cpds gun, car, soldier" => "unknown",
39   "in cpds post, town, zone" => "unknown",
40   "indef art" => "art",
41   "interj" => "itj",
42   "interj fire" => "itj",
43   "interj old" => "itj",
44   "interrog adj inv" => "unknown",
45   "interrog adv" => "unknown",
46   "interrog, rel adv" => "unknown",
47   "m" => "nn",
48   "m or" => "nn",
49   "n" => "nn",
50   "n see adj" => "nn",
51   "n also" => "nn",
52   "n inf see adj" => "nn",
53   "n pl" => "nn",
54   "n see adj" => "nn",
55   "n sing or pl" => "nn",
56   "n usu pl" => "nn",
57   "n, interj" => "nn",
```

```

58 "n, pron, adv" => "nn",
59 "no art" => "nn",
60 "no art, inv" => "nn",
61 "nt" => "nn",
62 "nt or" => "nn",
63 "pl" => "unknown",
64 "poss adj" => "adj",
65 "pref" => "unknown",
66 "pref brother, mother etc" => "unknown",
67 "pref in cpd vbs" => "unknown",
68 "pref mit n" => "unknown",
69 "prep" => "prep",
70 "pret, ptp" => "unknown",
71 "pret, ptp ~ vt" => "unknown",
72 "pron" => "pron",
73 "rel adv" => "unknown",
74 "suf" => "unknown",
75 "v" => "v",
76 "v aux" => "va",
77 "vb suf" => "v",
78 "vi" => "vi",
79 "vi, vt sep" => "vi",
80 "vt always separate" => "vt",
81 "vt sep and insep" => "vt",
82 "vt sep auch irreg" => "vt",
83 "vt usu pass" => "vt",
84 "vt, vt impers" => "vt",
85 "weak form" => "unknown"}
86
87 if ARGV.length != 3
88     STDERR.puts "Usage: mapper <lex-file> <german-out> <english-out>"
89     exit 1
90 end
91 if ARGV[0] == "-"
92     infile = STDIN
93 else
94     infile = open(ARGV[0], "r")
95 end
96 gfile = open(ARGV[1], "w")
97 efile = open(ARGV[2], "w")
98
99 count = 0
100 stat = {}
101 while line = infile.gets
102     count += 1
103     tokens = line.split('\t')
104     if tokens.length != 3
105         STDERR.puts "Illegal line #{count}"
106     end
107     lemma = tokens[0]
108     tag = tokens[1]
109     if lemma[-1,1] == '-' || lemma[0,1] == '-'
110         STDERR.puts "Deleting token '#{lemma}'"
111         next
112     end
113     if !hash.has_key?(tag)
114         STDERR.puts "Illegal Tag '#{tag}' in line #{count}"
115         next
116     end
117     pos = hash[tag]
118     if !stat.has_key?(pos)

```

```
119         stat[pos] = 0
120     end
121     stat[pos] += 1
122     tran = tokens[2]
123     gfile.print lemma, "\t", pos, "\t", lemma, "\n"
124     efile.print tran.strip, "\n"
125 end
126
127 infile.close unless infile == STDIN
128 stat.sort.each {|k, v| puts "#{k}: #{v}"}
```

Diverse Stringmethoden: string-utils.rb

```
1 =begin
2   string-utils.rb
3   div. String-Methoden
4   Arno Erpenbeck
5   KOKS Projekt, April/Mai 2001
6 =end
7
8 class String
9   def is_key?
10      return self[-1,1] == ':'
11   end
12
13   def is_brace?
14      return self[-1,1] == '('
15   end
16
17   def is_end?
18      return self[-1,1] == ')'
19   end
20
21   def is_start?
22      return self[0,1] == '('
23   end
24
25   def count_braces
26      count = 0
27      (0..self.length).each { |idx|
28         if self[idx,1] == '('
29            count += 1
30         elsif self[idx,1] == ')'
31            count -= 1
32         end
33      }
34      return count
35   end
36
37   def clear(extra = 0)
38      del = self.count_braces
39      if del >= 0
40         copy = self
41      else
42         del += extra
43         copy = self[0,self.length+del]
44      end
45      return copy
46   end
47 end
```

Kapselt Übersetzungsdaten aus Lexikon: translation.rb

```
1 =begin
2   translation.rb
3   kapselt Übersetzungsdaten aus Lexikon
4   Arno Erpenbeck
5   KOKS Projekt, April/Mai 2001
6 =end
7
8 class Translation
9
10     def initialize(orig=nil)
11         @orig = []
12         @orig << orig if orig
13         @pos = []
14         @tran = []
15     end
16
17     def reset_pos
18         @pos = []
19     end
20
21     def reset_tran
22         @tran = []
23     end
24
25     def fix_tran
26         @tran.last << ")" unless @tran.last[-1,1] == ')'
27     end
28
29     def orig=(what)
30         @orig << what
31     end
32
33     def pos=(what)
34         @pos << what
35     end
36
37     def tran=(what)
38         @tran << what
39     end
40
41     def dump
42         @orig.each {|x| print x, " "}
43         print "[ "
44         @pos.each {|x| print x, " "}
45         print "]"
46         @tran.each {|x| print x, " "}
47         print "\n"
48     end
49 end
50
51
```

Anhang K

Normalisierung – Code

K.1 normalize.py

Verwandelt Dateien aus vielen verschiedenen Formaten ins KoKS-Format.

```
1  #!/usr/bin/python
2  from htmllib import *
3  from sgmlib import *
4  from formatter import *
5  import sys, string, os, re
6
7  #TODO: bei SGML $xxx wegschmeissen ???
8  #TODO: Satzzeichen bei SGML!
9
10 class ToAscii:
11
12     def __init__(self, file, format=""):
13         self.infile = file
14         if format:
15             (self.outfile, self.done) = getfile(self.infile, format)
16         else:
17             self.outfile = file
18             self.done = 1
19
20     def convert(self):
21         return self.outfile
22
23 class Html2Ascii(ToAscii, HTMLParser):
24
25     def __init__(self, file):
26         ToAscii.__init__(self, file, "asc")
27         if not self.done:
28             self.fd = open(self.outfile, "w")
29             HTMLParser.__init__(self, AbstractFormatter(DumbWriter(self.fd)))
30
31     def convert(self):
32         if not self.done:
33             sys.stderr.write("Html2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
34             HTMLParser.feed(self, open(self.infile).read())
35             self.close()
36             self.fd.close()
37             return self.outfile
38
39     def anchor_end(self):
40         pass
41
42     def do_hr(self, attrs):
```

```
43     self.formatter.end_paragraph(2)
44
45 def do_img(self, attrs):
46     pass
47
48 def do_li(self, attrs):
49     HTMLParser.do_li(self, attrs)
50     self.formatter.add_line_break()
51
52 def do_p(self, attrs):
53     HTMLParser.do_p(self, attrs)
54     self.formatter.writer.send_literal_data("\n<ABSATZ>\n")
55
56 class Sgml2Ascii(ToAscii, SGMLParser):
57     #TODO: $xxx wegschmeissen ???
58     #TODO: Satzzeichen!!!
59
60     def __init__(self, file):
61         ToAscii.__init__(self, file, "asc")
62         if not self.done:
63             SGMLParser.__init__(self)
64         self.has_content = 0
65         self.has_dot = 0
66         self.is_head = 0
67         self.punktreg = re.compile('.*[!\?:\.\.]$', re.DOTALL)
68
69     def handle_data(self, data):
70         data = string.strip(data)
71         if data:
72             if self.is_head:
73                 self.out.write("\n")
74                 self.out.write(data + " ")
75                 self.has_content = 1
76                 if self.punktreg.match(data):
77                     self.has_dot = 1
78             elif self.is_head:
79                 self.out.write(".\n")
80                 self.has_dot = 1
81
82     # def start_head(self, args):
83     #     pass
84
85     # def end_head(self):
86     #     self.out.write("\n<ABSATZ>\n")
87
88     def start_s(self, args):
89         self.has_dot = 0
90
91     def end_s(self):
92         if not self.has_dot:
93             self.out.write(" .\n")
94         self.has_content = 1
95
96     def start_p(self, args):
97         self.has_dot = 0
98         self.out.write('\n')
99         if ('type', 'head') in args:
100             self.is_head = 1
101
102     def end_p(self):
103         self.is_head = 0
```

```
104
105 def start_h1(self, args):
106     if self.has_content:          # kein Delimiter vor erster Ueberschrift
107         self.out.write("\n<ABSATZ>\n")
108
109 def end_h1(self):
110     self.out.write("\n<ABSATZ>\n")
111     self.has_content = 0
112
113 def convert(self):
114     if not self.done:
115         sys.stderr.write("Sgml2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
116         self.out = open(self.outfile, "w")
117         SGMLParser.feed(self, open(self.infile).read())
118         self.out.close()
119         self.close()
120     return self.outfile
121
122 class Pdf2Ascii(ToAscii):
123
124     def __init__(self, file):
125         ToAscii.__init__(self, file, "html")
126
127     def convert(self):
128         if not self.done:
129             sys.stderr.write("Pdf2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
130             os.system("/usr/local/bin/pdf2html '%s' '%s'" %(self.infile, self.outfile))
131             #sys.stderr.write("Pdf2Ascii: converting %s from html...\n" %self.outfile)
132             h2a = Html2Ascii(self.outfile)
133             ascfile = h2a.convert()
134             return ascfile
135         return self.outfile
136
137 class Ps2Ascii(ToAscii):
138
139     def __init__(self, file):
140         ToAscii.__init__(self, file, "asc")
141
142     def convert(self):
143         if not self.done:
144             sys.stderr.write("Ps2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
145             os.system("/usr/bin/ps2ascii '%s' '%s'" %(self.infile, self.outfile))
146         return self.outfile
147
148 class Word2Ascii:
149
150     def __init__(self, file):
151         ToAscii.__init__(self, file, "asc")
152
153     def convert(self):
154         if not self.done:
155             sys.stderr.write("Word2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
156             os.system("/usr/bin/catdoc -a '%s' '%s'" %(self.infile, self.outfile))
157         return self.outfile
158
159 class Asc2Asc(ToAscii):
160     def __init__(self, file):
161         ToAscii.__init__(self, file, "asc2")
162
163     def convert(self):
164         if not self.done:
```



```

165 sys.stderr.write("Asc2Ascii: converting %s -> %s\n" %(self.infile, self.outfile))
166 input = open(self.infile, "r")
167 output = open(self.outfile, "w")
168 #     nlcount = 0
169 #     emptyline = 0
170 #     absatz = 0
171 for line in input.readlines():
172 #     if string.strip(line) == "":
173 #         emptyline = 1
174 #     else:
175 #         emptyline = 0
176 #         if string.find(line, "<ABSATZ>") >= 0:
177 #             absatz = 1
178 #         if not emptyline:
179 #             if nlcount:
180 #                 if not absatz:
181 #                     output.write("\n<ABSATZ>\n")
182 #                     nlcount = 0
183 #                     absatz = 0
184 #             else:
185 #                 nlcount = nlcount + 1
186
187 # convert to regular quotes
188 line = re.sub("<<", "\",", line)
189 line = re.sub(">>", "\",", line)
190 output.write(line)
191 # Das scheint irgendwie falsch
192 #if line[-9:] != "<ABSATZ>\n": # keine leeren Absaetze erzeugen
193 #     output.write("\n<ABSATZ>\n")
194 # tree-tagger-bug - workaround
195 output.write("\n<dokende>\nDas Dokument ist nun wirklich zu Ende.\n")
196 output.close()
197 input.close()
198 return self.outfile
199
200 def conv2ascii(file, format):
201     c2a = None
202     if format == "html":
203         c2a = Html2Ascii(file)
204     elif format == "sgml":
205         c2a = Sgml2Ascii(file)
206     elif format == "pdf":
207         c2a = Pdf2Ascii(file)
208     elif format == "ps":
209         c2a = Ps2Ascii(file)
210     elif format == "doc":
211         c2a = Word2Ascii(file)
212     elif format == "plain":
213         c2a = ToAscii(file)
214     else:
215         return None
216     a2a = Asc2Asc(c2a.convert())
217     return a2a.convert()
218 #     return c2a.convert()
219
220 def getfile(file, end, cutoff=1):
221     if(cutoff):
222         newfile = file[:string.rfind(file, ".")] + "." + end
223     else:
224         newfile = file + "." + end
225     done = 0

```

```

226     try:
227         open(newfile)
228         done = 1
229     except:
230         pass
231     return (newfile, done)
232
233 def main():
234     if len(sys.argv) < 3:
235         print "Usage: normalize.py <file> <format>"
236         print "         <format> is one of html,sgml,pdf,ps,doc,plain"
237         sys.exit(1)
238     conv2ascii(sys.argv[1], sys.argv[2])
239
240 if __name__ == '__main__':
241     main()
242
243

```

K.2 preproc.py

Normalisiert Korpora und trägt sie in die Datenbank ein.

```

1  #!/usr/bin/python
2
3  # $Header: /opt/cvs/bin/preproc.py,v 1.48 2001/08/05 14:38:35 koks Exp $
4  #
5  # preproc.py
6
7  from xmllib import *
8  import sys, getopt, string, os
9  from DatabaseAPI import config
10 import korpus2db, normalize
11
12 class SystemCall:
13
14     def __init__(self, file, end, cutoff=1):
15         self.infile = file
16         (self.outfile, self.done) = normalize.getfile(self.infile, end, cutoff)
17
18     def call(self, cmd, args, name):
19         if not self.done:
20             sys.stderr.write("%s %s\n" %(name, self.infile))
21             print cmd, args
22             os.system(cmd + " " + args)
23             return self.outfile
24
25 class InsertDelimiters:
26
27     def __init__(self, file):
28         self.infile = file
29         (self.outfile, self.done) = normalize.getfile(self.infile, "del")
30
31     def insdels(self, soft, sign):
32         if not self.done:
33             print "inserting delimiters..."
34             file = open(self.infile, "r")
35             out = open(self.outfile, "w")
36             for line in file.readlines():
37                 delim_written = 0
38                 if string.find(line, sign) > -1:

```

```

39         out.write(line)
40         out.write(soft)
41     elif string.find(line, "<ABSATZ>") > -1 and not delim_written:
42         out.write(soft)
43         out.write(line)
44     else:
45         out.write(line)
46         delim_written = 0
47     file.close()
48     out.close()
49     return self.outfile
50
51

```

```

52 class MapParser(XMLParser):
53

```

```

54     def __init__(self, file, kind):
55

```

```

56         XMLParser.__init__(self)
57         self.file = file
58         if string.rfind(file, "/") > -1:
59             self.dir = file[:string.rfind(file, "/")] + "/"
60         else:
61             self.dir = ""
62         self.use = 0
63         self.kind = kind
64

```

```

65     def parse_mapping(self):
66

```

```

67         self.mapping = []
68         self.nextdata = None
69         print "parsing index file " + self.file
70         self.feed(open(self.file).read())
71         self.close()
72

```

```

73     def start_mapping(self, attrs):
74

```

```

75         pass
76

```

```

77     def end_mapping(self):
78

```

```

79         pass
80

```

```

81     def start_document(self, attrs):
82

```

```

83         self.idx = len(self.mapping)
84         if not attrs.has_key("use") or attrs["use"] == "yes":
85             self.use = 1
86         else:
87             self.use = 0
88         if self.kind and attrs.has_key("kind") and attrs["kind"] != self.kind:
89             self.use = 0
90         if self.use:
91             self.mapping.append({})
92

```

```

93     def end_document(self):
94

```

```

95         self.end_info()
96

```

```

97     def start_part(self, attrs):
98

```

```

99         if self.use:
100             self.lang = attrs["lang"]
101             self.mapping[self.idx][self.lang] = {}
102             tagset = "ims"
103             if attrs.has_key("tagset"):
104                 tagset = attrs["tagset"]
105             self.mapping[self.idx][self.lang]["tagset"] = "tagset_" + tagset

```

```
100
101
102     def end_part(self):
103         pass
104
105     def start_file(self, attrs):
106         pass
107
108     def end_file(self):
109         pass
110
111     def start_filename(self, attrs):
112         if self.use:
113             self.nextdata = "file"
114             self.mapping[self.idx][self.lang]["format"] = attrs["format"]
115
116     def end_filename(self):
117         self.nextdata = None
118
119     def start_info(self, attrs):
120         pass
121
122     def end_info(self):
123         if self.use:
124             for key in ['author', 'comment', 'date', 'source', 'sourcetype']:
125                 if not self.mapping[self.idx].has_key(key):
126                     self.mapping[self.idx][key] = '?'
127
128     def start_author(self, attrs):
129         if self.use:
130             self.nextdata = "author"
131
132     def end_author(self):
133         self.nextdata = None
134
135     def start_comment(self, attrs):
136         if self.use:
137             self.nextdata = "comment"
138
139     def end_comment(self):
140         self.nextdata = None
141
142     def start_date(self, attrs):
143         if self.use:
144             self.nextdata = "date"
145
146     def end_date(self):
147         self.nextdata = None
148
149     def start_source(self, attrs):
150         if self.use:
151             self.mapping[self.idx]["sourcetype"] = attrs["type"]
152             self.nextdata = "source"
153
154     def end_source(self):
155         self.nextdata = None
156
157     def handle_data(self, data):
158         if self.use:
159             value = string.strip(data)
160             if not value:
```

```

161     value = "?"
162     if self.nextdata == "comment":
163         self.mapping[self.idx]["comment"] = value
164     elif self.nextdata == "file":
165         self.mapping[self.idx][self.lang]["file"] = self.dir + value
166     elif self.nextdata == "author":
167         self.mapping[self.idx]["author"] = value
168     elif self.nextdata == "date":
169         self.mapping[self.idx]["date"] = value
170     elif self.nextdata == "source":
171         self.mapping[self.idx]["source"] = value
172
173 class Enter2Db:
174
175     def __init__(self):
176         self.kdb = korpus2db.Korpus()
177         self.langmap = {'de': 1, 'en' : 2}
178
179     def enter2db(self, info):
180         file_de = info['de']['tagfile'] + ".al"
181         file_en = info['en']['tagfile'] + ".al"
182         (f1, done1) = normalize.getfile(file_de, "db")
183         (f2, done2) = normalize.getfile(file_en, "db")
184         if done1 or done2:
185             print "texts %s %s already entered into db..." %(file_en, file_de)
186             self.kdb.close()
187             return
188         qid = self.kdb.enterquelle(info['source'], info['sourcetype'], info['comment'])
189         herk = self.kdb.enterherkunft(info['de']['file'], info['author'], info['date'], qid)
190         print qid, herk
191         try:
192             print "entering texts into database..."
193             self.kdb.txt2db(herk, (file_de, "de", info['de']['tagset']), (file_en, "en", info['en']['tagset']))
194             self.kdb.close()
195             for i in (f1, f2):
196                 file = open(i, "w")
197                 file.close()
198         except IOError:
199             print "no aligned files found!"
200     #     except:
201     #         print "something bad happened while trying to enter %s" %info
202
203     def usage():
204         print "Usage: preproc.py [-c|-n] [-a] [-i <id>] [-k <kind>] <xml-files> ..."
205         print "-a          skip alignment"
206         print "-c          only convert"
207         print "-i <id>    run under different id (use with care)"
208         print "-k <kind>  only process entries of specified kind"
209         print "-n          do not enter stuff into database"
210         sys.exit()
211
212     def main():
213         # process command line args
214         try:
215             opts, args = getopt.getopt(sys.argv[1:], "achni:k:", ["align", "convert", "help", "nodb", "id=", "kind="])
216         except getopt.GetoptError:
217             # print help information and exit:
218             usage()
219         opt_id = None
220         opt_kind = None
221         opt_align = 1

```

```

222     opt_convert = 0
223     opt_enterdb = 1
224     for o, a in opts:
225         if o in ("-a", "--align"):
226             opt_align = 0
227         if o in ("-c", "--convert"):
228             if not opt_enterdb:
229                 usage()
230             opt_convert = 1
231         if o in ("-n", "--nodb"):
232             if opt_convert:
233                 usage()
234             opt_enterdb = 0
235         if o in ("-h", "--help"):
236             usage()
237         if o in ("-i", "--id"):
238             opt_id = a
239         if o in ("-k", "--kind"):
240             opt_kind = a
241     if len(args) < 1:
242         usage()
243
244     # flag -i should only be used together with -n
245     if opt_id and opt_enterdb:
246         sys.stderr.write("Warning: Using different id than dbname!\n")
247
248     # setup runfile name
249     if opt_id:
250         runfile = "/tmp/preproc_" + opt_id + ".running"
251     else:
252         runfile = "/tmp/preproc_" + config.dbname + ".running"
253
254     # check if already running
255     try:
256         file = open(runfile, "r")
257         file.close()
258         sys.exit("preproc.py is already running!")
259     except IOError:
260         pass
261     file = open(runfile, "w")
262     file.close()
263
264     langmap = {"en" : "english", "de": "german"}
265
266     #for i in sys.argv[start:]:
267     al_empty_err = []
268     for i in args:
269         xmlp = MapParser(i, opt_kind)
270         xmlp.parse_mapping()
271         for map in xmlp.mapping:
272             for lang in ["de", "en"]: #{
273                 if not map.has_key(lang):
274                     break
275                 ldict = map[lang]
276                 file = ldict["file"]
277                 if not os.access(file, os.R_OK):
278                     sys.stderr.write("File not found: %s\n" %file)
279                     break
280                 format = ldict["format"]
281                 if format != "tagged":
282                     ascfile = normalize.conv2ascii(file, format)

```

```

283     if opt_convert:
284         continue
285     if (ascfile):
286         a2t = SystemCall(ascfile, "tag")
287         pretagfile = a2t.call(getattr(config, "tagger_" + lang), " '%s' > '%s'" %(a2t.infile, a2t.outfile))
288         ldict["tagfile"] = pretagfile
289         #if(pretagfile):
290             # insdel = InsertDelimiters(pretagfile)
291             # ldict['tagfile'] = insdel.insdels("<SATZ>\n<segmentgrenze>\n", langmap[lang + "_si"])
292             # insdel = SystemCall(pretagfile, "del")
293             # ldict['tagfile'] = insdel.call("punkt-tagger-%s" %langmap[lang],
294             #                               " < '%s' > '%s' " %(insdel.infile, insdel.outfile),
295             #                               "inserting delimiters into ")
296     else:
297         if opt_convert:
298             sys.stderr.write("%s: nothing to do, because format = %s\n" %(file, format))
299             break # ?continue?
300         ldict["tagfile"] = ldict["file"]
301         sys.stderr.flush()
302     if(map.has_key('de') and map.has_key('en')): #{
303         de_file = None
304         if map['de'].has_key('tagfile'):
305             de_file = map['de']['tagfile']
306         en_file = None
307         if map['en'].has_key('tagfile'):
308             en_file = map['en']['tagfile']
309         if de_file and en_file and opt_align:
310             al = SystemCall(de_file, "al", 0)
311             #al.call(config.align, " -e -D '<dokende>' -d '<ABSATZ>' '%s' '%s'" %(de_file, en_file), "al")
312             al.call(config.align, " -e -D '<ABSATZ>' -d '<segmentgrenze>' '%s' '%s'" %(de_file, en_file))
313             if opt_enterdb:
314                 quit = 0
315                 for f in(de_file, en_file):
316                     al_file = f + ".al"
317                     if not os.access(al_file, os.R_OK) or \
318                         not string.strip(open(al_file, "r").read()):
319                         #al_file.close()
320                         al_empty_err.append(f)
321                         quit = 1
322                 if quit:
323                     continue
324                 print "enter2db"
325                 e2d = Enter2Db()
326                 e2d.enter2db(map)
327             sys.stderr.flush()
328         #}
329     if al_empty_err:
330         print "!!!empty aligned files:", al_empty_err, "!!!"
331     os.unlink(runfile)
332
333 main()
334

```

K.3 py2pl.py

Macht die Python-Konfigurationsdateien für Perl nutzbar.

```

1 #!/usr/bin/python
2
3 # $Header: /opt/cvs/bin/py2pl.py,v 1.7 2001/08/04 11:58:40 koks-cvs Exp $
4 #

```

```

5 # read config.py and write out data in perl-usable style
6 #
7 # KOKS project
8 # 31.07.2001
9
10 from DatabaseAPI import config
11 import sys
12
13 # TODO
14 # - fix error with arrays/hasches
15
16 print """
17 #!/usr/bin/perl
18
19 # config file generated by py2pl
20
21 use strict;
22 use File::stat;
23
24 # testen, ob config.py neuer ist als config.pl
25 my $dir = `dirname $0`;
26 chomp($dir);
27 my $pyfile;
28 if (defined(%ENV->{PYTHONPATH})) {
29     my @paths = split(/:/, %ENV->{PYTHONPATH});
30     my $path = "";
31     foreach $path (@paths) {
32         $pyfile = $path . "/DatabaseAPI/config.py";
33         open FOO, $path and last;
34     }
35     close FOO;
36 } else {
37     $pyfile = $dir . "/config.py";
38 }
39
40 if (stat($dir . "/config.pl")->mtime < stat($pyfile)->mtime) {
41     print STDERR "Warning: config.pl is older than config.py!\n";
42 }
43
44 {
45     """
46     print 'dbname => "%s",' %(config.dbname)
47     print 'dbtype => "%s",' %(config.dbtype)
48     print 'dbpass => "%s",' %(config.dbpass)
49     print 'dbuser => "%s",' %(config.dbuser)
50     print 'qdb_ports => %s,' %(config.qdb_ports)
51     print 'qdb_host => "%s",' %(config.qdb_host)
52     print 'align => "%s",' %(config.align)
53     print 'al_jar => "%s",' %(config.al_jar)
54     print 'tags => "%s",' %(config.tags)
55     print 'databaseapi => "%s",' %(config.databaseapi)
56     print 'tagger_de => "%s",' %(config.tagger_de)
57     print 'tagger_en => "%s",' %(config.tagger_en)
58     print 'wbsize => "%s",' %(config.wbsize)
59     print 'wbids => "%s",' %(config.wbids)
60     print 'goodcorpora => "%s",' %(config.goodcorpora)
61     print '#wbdateien => %s' %(config.wbdateien)
62     print '}'

```


Anhang L

Tagging – Code

L.1 tree-tagger-Skripte

L.1.1 tree-tagger-english

```
#!/bin/sh

# Set these paths appropriately
#HOME=$HOME/koks
[ -z $TAGGERHOME ] && TAGGERHOME=/home/koks
BIN=$TAGGERHOME/tagger/bin
CMD=$TAGGERHOME/tagger/cmd
LIB=$TAGGERHOME/tagger/lib
KOKSBIN=`dirname "$0"`

TOKENIZER=${BIN}/separate-punctuation
CORRECTION=${KOKSBIN}/metatag-correction
TAGGER=${BIN}/tree-tagger
ABBR_LIST=${LIB}/english-abbreviations
PARFILE=${LIB}/english.par
SENT_REC=${KOKSBIN}/punkt-tagger-english
SENT_REC_FIX=${KOKSBIN}/punkt.py          # Bug-Fix 21. Juni 2001

# put all on one line
cat $* |
# do tokenization
$TOKENIZER +1 +s +1 $ABBR_LIST |
# separate clitics from preceding words
sed -e "s/'s''$'/'s/g" \
-e "s/s''$'/'/g" \
-e "s/n't''$'/'n't/g" \
-e "s/'re''$'/'re/g" \
-e "s/'ve''$'/'ve/g" \
-e "s/'d''$'/'d/g" \
-e "s/'m''$'/'m/g" \
-e "s/'em''$'/'em/g" \
-e "s/'ll''$'/'ll/g" \
-e '/^$/d' |
tr ' ' '\n' |
# correct metatags
$CORRECTION |
# remove empty lines
grep -v '^$' |
# tagging
$TAGGER $PARFILE -token -lemma -sgml |
#$TAGGER $PARFILE -token -lemma -sgml
# recognize sentence endings
```

```
$SENT_REC |
$SENT_REC_FIX
```

L.1.2 tree-tagger-german

```
#!/bin/sh

# Set these paths appropriately
#HOME=$HOME/koks
[ -z $TAGGERHOME ] && TAGGERHOME=/home/koks

BIN=$TAGGERHOME/tagger/bin
CMD=$TAGGERHOME/tagger/cmd
LIB=$TAGGERHOME/tagger/lib
KOKSBIN=$TAGGERHOME/bin

TOKENIZER=${BIN}/separate-punctuation
CORRECTION=${KOKSBIN}/metatag-correction
GERMANIZER="perl -I${KOKSBIN} ${KOKSBIN}/umlaut.pl"
TAGGER=${BIN}/tree-tagger
ABBR_LIST=${LIB}/german-abbreviations
PARFILE=${LIB}/german-big.par
#PARFILE=${LIB}/german-small.par
FILTER=${CMD}/filter-german-tags
SENT_REC=${KOKSBIN}/punkt-tagger-german
SENT_REC_FIX=${KOKSBIN}/punkt.py          # Bug-Fix 21. Juni 2001
CLEANUP=${KOKSBIN}/cleanup.rb

# put all on one line
cat $* |
# do tokenization
$TOKENIZER +l +s +l $ABBR_LIST |
# correct metatags
$CORRECTION |
# remove empty lines
grep -v '^$' |
# add Umlaut where necessary
$GERMANIZER |
# tagging
$TAGGER $PARFILE -token -lemma -sgml |
# error correction
$FILTER |
# recognize sentence endings
$SENT_REC |
$SENT_REC_FIX
```

L.2 Sonderzeichenrekonstruktion

```
#!/usr/bin/perl -w

use strict;
use DBI;

# $Header: /opt/cvs/bin/umlaut.pl,v 1.12 2001/08/03 12:03:41 koks-cvs Exp $
#
# umlaut.pl
#
# Output vom Tokenizer lesen und dabei versuchen, "ehemalige" Umlaute
# zu rekonstruieren.
```

```

# Benutzt Datenbank und ein bisschen Linguistik...

# lese Konfigurationsdatei ein
# config.pl wird mit py2pl aus config.py erzeugt :-)
my $config = require("config.pl");

my $user=$config->{dbuser};
my $passwd=$config->{dbpass};
my $dbname="DBI:mysql:dbname=" . $config->{dbname};

my $dbh = DBI->connect($dbname, $user, $passwd) || die("No database connection!");

while(my $word=<STDIN>){
    my $result = ersetze($word);
    print "$result\n";
}

$dbh->disconnect();

sub ersetze{
    my $word = $_[0];
    chomp $word;
    my $output = $word;
    my $known_in_db = 0;

    # unbekanntes Wort mit Umlaut/ß-Kandidaten?
    if (($word =~ /[aou|ss]/i)){
        # aufsplitten in Teile mit nur einem ([aou|ss)
        $word =~ s/([aou|ss)]/$1\t/ig;
        # $word =~ s/([e][aou|ss)]/$1\t/ig;
        my @word = split /\t/, $word;
        my $length = @word;
        my $last = pop @word;

        # letztes Element ohne ([aou|ss)? -> mit vorletztem vereinen
        if (($last !~ /[aou|ss]/i) && ($length > 1)){
            push @word, (pop @word) . $last;
            $length--;
        }
        else{
            push @word, $last;
        }

        # alle Kandidaten generieren
        my @candidates = genAll(@word);
        my $cand_len = @candidates;

        #   for(my $j = $cand_len-1; $j >= 0; $j--){
        #       if (defined(dblookup($candidates[$j]))){
        #           $output = $candidates[$j];
        #       }
        #   }

        foreach (@candidates){
            if (defined(dblookup($_))){
                $known_in_db = 1;
                $output = $_;
                last;
            }
        }
    }
}

```

```

# für den Fall, daß das Wort unbekannt ist: alles ersetzen (außer ss)
if($known_in_db == 0){
  $output =~ s/Ae/Ä/g;
  $output =~ s/Oe/Ö/g;
  $output =~ s/Ue/Ü/g;
  $output =~ s/ae/ä/g;
  $output =~ s/oe/ö/g;
  #$output =~ s/ue/ü/g;
  $output =~ s/([^aeqQ])ue/$1ü/g; # 'erneuern' != 'erneürn', 'Erbauer' != 'Erbaür', 'Quelle' != 'Qülle'
  my $rest = $output;
  my $anfang = "";
  # ss nach Diphtong zu ß außer an Kompositumsgrenze
  while($rest =~ /(au|ei|aeu|äu|eu)ss/){
    my @split = split //, $rest;
    while (($anfang !~ /s$/) || ($split[0] ne "s")){
      $anfang .= shift @split;
    }
    $rest =~ s/$anfang(.*)$/$1/g;
    # Kompositumsgrenze
    $rest = ersetze($rest);
    if(!defined(dblookup($rest))){
      if ($rest !~ /^s[cpt]/){
        $anfang =~ s/(.*)s/$1ß/g;
        $rest =~ s/s(.*)$/$1/g;
      }
    }
  }
  $output = $anfang . $rest;
}

# Rückgabe
return $output;
}

sub genAll{
  my @parts = @_;
  my @result = ();
  my $len = @parts;
  my $first = shift @parts;
  my $first2 = $first;
  my $subs = ($first =~ s/ae/ä/);
  $subs = ($first =~ s/Ae/Ä/) if !$subs;
  $subs = ($first =~ s/oe/ö/) if !$subs;
  $subs = ($first =~ s/Oe/Ö/) if !$subs;
  $subs = ($first =~ s/ue/ü/) if !$subs;
  $subs = ($first =~ s/Ue/Ü/) if !$subs;
  $subs = ($first =~ s/ss/ß/) if !$subs;
  if($len == 1){
    push @result, $first2;
    push @result, $first;
  }
  else{
    my @rest = genAll(@parts);
    foreach (@rest){
      push @result, $first2 . $_;
      push @result, $first . $_;
    }
  }
  return @result;
}

```

```

sub dblookup{
  my $text = $_[0];
  $_ = $text;
  s/'''/';
  my $sth = $dbh->prepare("select t.name from tokens t, grundformen g where t.name = '" . $_ . "' and g.name <
  $sth->execute();
  my $result = ($sth->fetchrow())[0];
  $sth->finish();
  return $result;
}

```

L.3 Korrektur der Metatags

```
#!/usr/bin/perl -w
```

```

while($metatag = <STDIN>){
  $dotted = 0;
  chomp $metatag;
  if($metatag =~ /^([\^\.])(\.\.+)$/){
    print "$1\n";
    $metatag = $2;
  }
  if($metatag =~ /^\.\.+$/){
    $dotted = 1;
    while( (defined($next=<STDIN>))
      && ($next =~ /^\.+$/))
    {
      $metatag .= $next;
      chomp $metatag;
    }
    print "$metatag\n";
  }
  if(!$dotted){
    while ($metatag =~ /<[^>]*$/){
      if(defined($next = <STDIN>)){
        $metatag = $metatag . " $next";
        chomp $metatag;
      }
    }
    print "$metatag\n";
  }
}

```

L.4 Satzendenerkennung

L.4.1 punkt-tagger-english

```
#!/usr/bin/perl -w
```

```

# #####
# Skript, das in der Ausgabe des englischen Tree-Taggers erkennt, #
# ob es sich bei einem als "satzbeendendes Interpunktionzeichen" #
# getaggten Zeichen tatsaechlich um einen solchen handelt; #
# erkennt auch Punkte nach Ordinalzahlen oder Abkuerzungen als solche #
# #

```

```

# Dieses Skript liegt auch fuer den deutschen Tree-Tagger vor           #
#                                                                       #
# liest von STDIN, schreibt nach STDOUT                                #
#                                                                       #
# #####                                                                #

my %tags;
$tags{'punkt'} = "SENT";
$tags{'unknown'} = "<unknown>";
$tags{'satz'} = "<SATZ>";
$tags{'segment'} = "<segmentgrenze>";
$tags{'absatz'} = "<ABSATZ>";
$tags{'ende'} = "<DOKENDE>";

# Initialisierung

defined($zwei=<STDIN>) || exit 0;
#while($zwei =~ /\s*</){
while($zwei !~ /\s*[\w_äöüÄÖÜß\.\!\/]){
    print "$tags{'satz'}\n$tags{'segment'}\n" if($zwei eq "$tags{'absatz'}");
    print $zwei;
    defined($zwei = <STDIN>) || exit 0;
}

print $zwei;                    # erste Zeile ausgeben, die wird mangels
                                # Kontext nicht ueberprueft

defined($drei=<STDIN>) || exit 0;
# while($drei =~ /\s*</){
while($drei !~ /\s*[\w_äöüÄÖÜß\.\!\/]){
    print "$tags{'satz'}\n$tags{'segment'}\n" if($drei eq "$tags{'absatz'}");
    print $drei;
    defined($drei = <STDIN>) || exit 0;
}

#####

# los geht's

while($neueZeile=<STDIN>) {      # neue Zeile einlesen

    # Schluss machen, wenn <dokende>
    $trash = $neueZeile;
    chomp($trash);
    if (lc($trash) eq lc($tags{'ende'})) {
        last;
    }

    $unveraendert = 1;
    $commentBuffer = "";

    $eins = $zwei;                # Zeilenfenster aktualisieren
    $zwei = $drei;
    $drei = $neueZeile;

    # Kommentarzeilen ausfiltern

    #while($drei =~ /\s*</){
    while($drei !~ /\s*[\w_äöüÄÖÜß\.\!\/]){
        $commentBuffer = $commentBuffer . "$tags{'satz'}\n$tags{'segment'}\n" if($drei eq "$tags{'absatz'}");
        $commentBuffer = $commentBuffer . $drei;
    }
}

```

```

if (!(defined($drei = <STDIN>))){
    print $commentBuffer;
    exit 0;
}
}

chomp ($eins, $zwei, $drei);

@eins = split /\t/, $eins;      # Zugriff auf Wort, Tag, Lemma schaffen
@zwei = split /\t/, $zwei;
@drei = split /\t/, $drei;

if($zwei[1] eq $tags{'punkt'}){      # wenn mittlere Zeile als Punkt getagt
    # Wort nach Punkt groß, Lemma klein -> Satzende

    if (    ($drei[2] !~ /^</)          # Lemma != <unknown>
        && (lcfirst($drei[0]) ne $drei[0]) # Wort groß
        && (lcfirst($drei[2]) eq $drei[2]) # Lemma klein
    )
    {
        print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
        next;
    }

    # Zahl vor Punkt, danach Monatsname oder anderes typisches Wort
    # -> Ordinalzahl

    if ($eins[0] =~ /[0-9]+/ && $monatsname{$drei[2]}){
        print "$zwei[0]\tORD-P\t$zwei[2]\n$commentBuffer";
        next;
    }

    # Wort vor Punkt ohne Vokal -> Abkuerzung

    if ($eins[0] !~ /[AEIOUÄÖÜYaeiouäöü]/){
        print "$zwei[0]\tABK-P\t$zwei[2]\n$commentBuffer";
        next;
    }

    # Wort vor Punkt hat unzulässigen absoluten Endrand -> Abkürzung
    # zulaessige engl. Endraender?

    # <ABSATZ>-Markierung bedeutet auch Satzgrenze

    if($zwei =~ /^<ABSATZ>$/i){
        print "$tags{'satz'}\n$tags{'segment'}\n$tags{'absatz'}\n$commentBuffer";
        next;
    }

    # Default: Satzgrenze

    if ($unveraendert){
        print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
        next;
    }
}

# Auch wenn kein abgerueckter Punkt:
# Wort nach Punkt groß, Lemma klein -> Abkuerzung am Satzende

if (($zwei[0] =~ /.+\.$/))

```



```

$tags{'punkt'} = "\$.";
$tags{'unknown'} = "<unknown>";
$tags{'satz'} = "<SATZ>";
$tags{'segment'} = "<segmentgrenze>";
$tags{'absatz'} = "<ABSATZ>";
$tags{'ende'} = "<DOKENDE>";

# Initialisierung

defined($zwei=<STDIN>) || exit 0;
#while($zwei =~ /\s*</){
while($zwei !~ /\s*[\w_äöüÄÖÜß\.]\/){
    print "$tags{'satz'}\n$tags{'segment'}\n" if ($zwei eq $tags{'absatz'});
    print $zwei;
    defined($zwei = <STDIN>) || exit 0;
}

print $zwei;                                # erste Zeile ausgeben, wird mangels
                                             # Kontext nicht ueberprueft

defined($drei=<STDIN>) || exit 0;
#while($drei =~ /\s*</){
while($drei !~ /\s*[\w_äöüÄÖÜß\.]\/){
    print "$tags{'satz'}\n$tags{'segment'}\n" if ($drei eq $tags{'absatz'});
    print $drei;
    defined($drei = <STDIN>) || exit 0;
}

#####

# los geht's
while($neueZeile=<STDIN>){                  # neue Zeile einlesen

    # Schluss bei <dokende>
    $trash = $neueZeile;
    chomp($trash);
    if (lc($trash) eq lc($tags{'ende'})) {
        #print STDERR "Hab Dich: $trash\n";
        last;
    }
    #print STDERR "Normal: $trash\n";

    # Zeilen mit Müll gleich wegwerfen
    #@trash = split /\t/, $trash;
    #if (($trash[2] eq $UNKNOWN) && (length($trash[0]) > 1)) {
        ##print STDERR "Tata '$trash[0]' -->";
        #if ($trash[0] !~ /[a-zA-ZäöüÄÖÜß0-9]+\/) {
            ##print STDERR "[-]\n";
            #next;
        #}
        ##print STDERR "[+]\n";
    #}

    $unveraendert = 1;
    $commentBuffer = "";

    $eins = $zwei;                          # Zeilenfenster aktualisieren
    $zwei = $drei;
    $drei = $neueZeile;

    # Kommentarzeilen ausfiltern

```

```

# while($drei =~ /^s*</)
while($drei !~ /^s*[\w_äüÄÖÜß\.\!\?]/){
    $commentBuffer = $commentBuffer . "$tags{'satz'}\n$tags{'segment'}\n" if($drei eq $tags{'absatz'});
    $commentBuffer = $commentBuffer . $drei;
    if (!(defined($drei = <STDIN>))){
        print $commentBuffer;
        exit 0;
    }
}

chomp ($eins, $zwei, $drei);

@eins = split /\t/, $eins;      # Zugriff auf Wort, Tag, Lemma schaffen
@zwei = split /\t/, $zwei;
@drei = split /\t/, $drei;

if($zwei[1] eq $tags{'punkt'}){      # wenn mittlere Zeile als Punkt getagt
    if($zwei[0] eq "."){

        # Wort nach Punkt groß, Lemma klein -> Satzende

        if ( ($drei[2] !~ /^</)          # Lemma != <unknown>
            && (lcfirst($drei[0]) ne $drei[0]) # Wort groß
            && (lcfirst($drei[2]) eq $drei[2]) # Lemma klein
        )
        {
            print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
            next;
        }

        # Zahl vor Punkt, danach Monatsname oder anderes typisches Wort
        # -> Ordinalzahl

        if ($eins[0] =~ /[0-9]+/ && $monatsname{$drei[2]}){
            print "$zwei[0]\tORD-P\t$zwei[2]\n$commentBuffer";
            next;
        }

        # Wort vor Punkt ohne Vokal -> Abkuerzung

        if( ($eins[0] !~ /[AEIOUÄÖÜYaeiouäöüy]/)
            && ($eins[0] =~ /\wß/))
        {
            print "$zwei[0]\tABK-P\t$zwei[2]\n$commentBuffer";
            next;
        }

        # Wort vor Punkt hat unzulässigen absoluten Endrand -> Abkürzung
        #
        # $endrand = $eins[0];
        # $endrand =~ s/^(.*([^aeiouäöüy]*))$/1/g;
        # if (! $endrand{$endrand}){
        #     print "$zwei[0]\tABK-P\t$zwei[2]\n";
        #     print $commentBuffer;
        #     $unveraendert = 0;
        #     next;
        # }
        #

```

<ABSATZ>-Markierung bedeutet auch Satzgrenze

```

if($zwei =~ /^<ABSATZ>$/i){
  print "$tags{'satz'}\n$tags{'segment'}\n$tags{'absatz'}\n$commentBuffer";
  next;
}

# Default: Satzgrenze

if ($unveraendert){
  print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
  next;
}
}
elseif ($zwei[0] eq "!" || $zwei[0] eq "?") {
  # Nach '?' und '!' immer Satzende
  print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
  next;
}

# if($zwei[0] eq ":"){
#   # Wort nach Punkt groß, Lemma klein -> Satzende
#   #
#   #   if (      ($dreier[2] !~ /^</)          # Lemma != <unknown>
#   #         && (lcfirst($dreier[0]) ne $dreier[0]) # Wort groß
#   #         && (lcfirst($dreier[2]) eq $dreier[2]) # Lemma klein
#   #       )
#   #   {
#   #     print "$zwei[0]\tSATZ-P\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
#   #     next;
#   #   }
#   #
#   #   # <ABSATZ>-Markierung bedeutet auch Satzgrenze
#   #
#   #   if($zwei =~ /^<ABSATZ>$/i){
#   #     print "$tags{'satz'}\n$tags{'segment'}\n$tags{'absatz'}\n$commentBuffer";
#   #     next;
#   #   }
# }

# Auch wenn kein abgerueckter Punkt:
# Wort nach Punkt groß, Lemma klein -> Abkuerzung am Satzende

if (($zwei[0] =~ /.+\.$/))
  && ($dreier[2] !~ /^</)
  && (lcfirst($dreier[0]) ne $dreier[0])
  && (lcfirst($dreier[2]) eq $dreier[2]))
{
  print "$zwei[0]\t$zwei[1]\t$zwei[2]\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
  $unveraendert = 0;
}

# nur Punkte (z.B. "...")

#if ($zwei[0] =~ /\.\.+$/){
#  # print "$zwei[0]\t$($zwei[2])\n$tags{'satz'}\n$tags{'segment'}\n$commentBuffer";
#  # $unveraendert = 0;
#}

# Ansonsten Zeile ausgeben

```

```

print "$zwei\n$commentBuffer" if ($unveraendert);
}

# Ganz zum Schluß: Letzte Zeile ausgeben

if($drei[1] eq $tags{'punkt'}){
  print "$drei[0]\tSATZ-P\t$drei[2]\n$tags{'satz'}\n$tags{'segment'}\n";
}
else{
  print "$drei\n$tags{'satz'}\n$tags{'segment'}\n";
}
print "$tags{'ende'}\n";

#####

# ein paar nuetzliche Hashes

%monatsname = ( 'Januar' => 1,
                'Februar' => 1,
                'März' => 1,
                'April' => 1,
                'Mai' => 1,
                'Juni' => 1,
                'Juli' => 1,
                'August' => 1,
                'September' => 1,
                'Oktober' => 1,
                'November' => 1,
                'Dezember' => 1,
                'Platz' => 1,
                'Preis' => 1,
                'Versuch' => 1,
                'Anlauf' => 1
                );

#%endrand = ('' => 1,
#           'b' => 1, 'bb' => 1, 'bbs' => 1, 'bbst' => 1, 'bs' => 1,
#           'bsch' => 1, 'bschs' => 1, 'bschst' => 1, 'bscht' => 1,
#           'bst' => 1, 'bsts' => 1,
#           'ch' => 1, 'chs' => 1, 'chst' => 1, 'ck' => 1, 'cks' => 1,
#           'ckst' => 1,
#           'd' => 1, 'dd' => 1, 'ds' => 1, 'dst' => 1,
#           'f' => 1, 'ff' => 1, 'ffs' => 1, 'ffst' => 1, 'fs' => 1, 'fst' => 1,
#           'g' => 1, 'gg' => 1, 'ggs' => 1, 'ggst' => 1, 'gs' => 1, 'gst' => 1,
#           'h' => 1,
#           'k' => 1, 'ks' => 1, 'kst' => 1,
#           'l' => 1, 'lb' => 1, 'lbs' => 1, 'lbst' => 1, 'lbt' => 1, 'ld' => 1,
#           'lds' => 1, 'ldst' => 1, 'lf' => 1, 'lfs' => 1, 'lfst' => 1,
#           'lft' => 1, 'lg' => 1, 'lgs' => 1, 'lgst' => 1, 'lgt' => 1,
#           'lk' => 1, 'lks' => 1, 'lkst' => 1, 'lkt' => 1, 'll' => 1,
#           'lls' => 1, 'llst' => 1, 'lm' => 1, 'lms' => 1, 'lmst' => 1,
#           'lmt' => 1, 'ln' => 1, 'lns' => 1, 'lnst' => 1, 'lnt' => 1,
#           'lp' => 1, 'lps' => 1, 'lpst' => 1, 'lpt' => 1, 'ls' => 1,
#           'lsch' => 1, 'lschs' => 1, 'lschst' => 1, 'lscht' => 1,
#           'lst' => 1, 'lt' => 1, 'lts' => 1, 'ltst' => 1, 'lz' => 1,
#           'lzt' => 1,
#           'm' => 1, 'mb' => 1, 'mbs' => 1, 'mbst' => 1, 'mbt' => 1, 'md' => 1,
#           'mds' => 1, 'mdst' => 1, 'mf' => 1, 'mfs' => 1, 'mfst' => 1,
#           'mft' => 1, 'mm' => 1, 'mms' => 1, 'mmst' => 1, 'mp' => 1,
#           'mpf' => 1, 'mpfs' => 1, 'mpftmpfst' => 1, 'mps' => 1,
#           'mpst' => 1, 'mpt' => 1, 'ms' => 1, 'mst' => 1, 'mt' => 1,

```

```

#         'mts' => 1, 'mtst' => 1, 'mz' => 1, 'mzt' => 1,
#         'n'  => 1, 'nd'  => 1, 'nds' => 1, 'ndst' => 1, 'ndt' => 1, 'nf' => 1,
#         'nfs' => 1, 'nfst' => 1, 'nft' => 1, 'ng'  => 1, 'ngs' => 1,
#         'ngst' => 1, 'ngt' => 1, 'nk'  => 1, 'nks' => 1, 'nkst' => 1,
#         'nkt' => 1, 'nn'  => 1, 'nns' => 1, 'nnst' => 1, 'ns'  => 1,
#         'nsch' => 1, 'nschs' => 1, 'nschst' => 1, 'nscht' => 1,
#         'nst'  => 1, 'nt'  => 1, 'nts' => 1, 'ntst' => 1, 'nz'  => 1,
#         'nzt'  => 1,
#         'p'  => 1, 'pf'  => 1, 'pfs' => 1, 'ppst' => 1, 'ps'  => 1, 'pt'  => 1,
#         'r'  => 1, 'rb'  => 1, 'rbs' => 1, 'rbst' => 1, 'rbsts' => 1,
#         'rbt' => 1, 'rd'  => 1, 'rds' => 1, 'rdst' => 1, 'rf'  => 1,
#         'rfs' => 1, 'rfst' => 1, 'rft' => 1, 'rg'  => 1, 'rgs' => 1,
#         'rgst' => 1, 'rgt' => 1, 'rk'  => 1, 'rks' => 1, 'rkst' => 1,
#         'rkt' => 1, 'rl'  => 1, 'rls' => 1, 'rm'  => 1, 'rms' => 1,
#         'rmst' => 1, 'rmt' => 1, 'rn'  => 1, 'rns' => 1, 'rnst' => 1,
#         'rnt' => 1, 'rp'  => 1, 'rps' => 1, 'rpst' => 1, 'rpt' => 1,
#         'rrst' => 1, 'rs'  => 1, 'rsch' => 1, 'rschs' => 1,
#         'rschst' => 1, 'rscht' => 1, 'rst'  => 1, 'rt'  => 1, 'rts' => 1,
#         'rtst' => 1, 'rz'  => 1, 'rzt'  => 1,
#         's'  => 1, 'sch' => 1, 'schs' => 1, 'schst' => 1, 'scht' => 1,
#         'sk'  => 1, 'ss'  => 1, 'sst' => 1, 'st'  => 1, 'sts' => 1,
#         't'  => 1, 'ts'  => 1, 'tt'  => 1, 'tts' => 1, 'tz'  => 1, 'tzt' => 1,
#         'v'  => 1,
#         'x'  => 1, 'xt'  => 1,
#         'z'  => 1, 'zt'  => 1
#);
#

```

Anhang M

Alignment – Code

M.1 Region Paket

Die Quellen des Python-Pakets Region befinden sich im CVS Baum im Verzeichnis align/Region.

M.1.1 __init__.py

```
1  #!/usr/bin/python
2
3  # Initialisierung des Pakets "Region"
4
5  __all__ = [
6      "absatz.py",
7      "atom.py",
8      "dokument.py",
9      "korpus.py",
10     "region.py",
11     "satz.py",
12     "segment.py"
13 ]
```

M.1.2 absatz.py

```
1  #!/usr/bin/python
2
3  # Modul Region.absatz
4
5  # KOKS Studienprojekt 2000 - 2001
6  # Korpusbasierte Kollokationensuche
7
8  import alignment
9  import region
10 import segment
11 import string
12
13 class Absatz(region.Regionen): #{
14
15     def __repr__(self):
16         rtext = self.getText()
17         rtext = string.replace(rtext, "\n", " ")
18         rtext = string.replace(rtext, "\t", " ")
19         if len(rtext) > 64:
20             rtext = rtext[:60] + "..."
21         return "<Absatz: %s>" %rtext
22
23     def getText(self):
24         return region.Regionen.getText(self) + "\n"
347
```

```

25
26 def getDelimiter(self):
27     return "<ABSATZ>\n"
28
29 delimiter = "<ABSATZ>\n"
30
31 def quelleninfo(self):
32     return "Absatz %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)
33
34 #}
35
36 class AbsatzPaar(Absatz, alignment.AlignedRegion): #{
37
38     def __init__(self, abs1, abs2, qs, zs):
39         alignment.AlignedRegion.__init__(self, abs1, abs2, qs, zs)
40         Absatz.__init__(self, abs1)
41
42     def alignedRegion(self): # liefert den alignnten Absatz
43         return AbsatzPaar(self.reg2, self.reg1, self.lang2, self.lang1)
44
45     def getItemAtIndex(self, index):
46         segment1 = self.reg1[index]
47         try:
48             segment2 = self.reg2[index]
49         except IndexError:
50             return self.reg1[index]
51         retval = segment.SegmentPaar(segment1, segment2, self.lang1, self.lang2)
52         retval.setQuelleninfo(index, self.quelleninfo())
53         return retval
54
55 #}
56
57 class AbsatzVonZeilenliste(Absatz): #{
58
59     def __init__(self, zeilenliste):
60         segmentliste = []
61         pos = 0
62         start = 0
63         inhaltGelesen = 0
64         for zeile in zeilenliste: #{
65             if zeile[0:15] == "<segmentgrenze>": #{
66                 segmentliste.append(
67                     segment.SegmentVonZeilenliste(zeilenliste[start:pos]))
68                 start = pos + 1
69                 inhaltGelesen = 0
70             #}
71             else: #{
72                 inhaltGelesen = 1
73             #}
74             pos = pos + 1
75         #}
76         if inhaltGelesen: #{
77             segmentliste.append(
78                 segment.SegmentVonZeilenliste(zeilenliste[start:pos]))
79         #}
80         Absatz.__init__(self, segmentliste)
81 #}
82
83 if __name__ == '__main__':
84     import atom
85     a = "Äpfel\tNN\tApfel\n"

```

```

86     b = "Birnen\tNN\tBirne\n"
87     c = "<segmentgrenze>\n"
88     d = "Cashewnüsse\tNN\tCashewnuß\n"
89     e = "Datteln\tNN\tDattel\n"
90     f = "<segmentgrenze>\n"
91     g = AbsatzVonZeilenliste([a,b,c,d,e,f,a,b,c])
92     print "Absatz [a,b,...,a,b,...]:" , g , `g`
93     r = g.getRegionenMit(segment.Segment)
94     print "Regionen mit Segment:" , r , `r` , len(r)
95     for i in range(0, len(r)):
96         print ("["+`i`+"]:" + `r[i]`)
97     r = g.getRegionenMit(atom.Atom)
98     print r , `r` , len(r)
99     for i in range(0, len(r)):
100        print ("["+`i`+"]:" + `r[i]`)

```

M.1.3 alignment.py

```

1  #!/usr/bin/python
2
3  # Modul Region.alignment
4
5  import sys
6  import types
7  import region
8
9  class AlignedRegion:
10
11     def __init__(self, reg1, reg2, lang1, lang2): # i.d.R. nicht verwendet
12         self.reg1 = reg1
13         self.reg2 = reg2
14         self.lang1 = lang1
15         self.lang2 = lang2
16
17     def alignedRegion(self):           # ueberschreiben
18         return None                   # liefert die Region, die zu self align ist
19
20     def getQuellsprache(self):
21         return "undefiniert"
22
23     def getZielsprache(self):
24         return "undefiniert"
25

```

M.1.4 atom.py

```

1  #!/usr/bin/python
2
3  import region
4  import string
5  import sys
6
7  class Atom(region.Region): #{
8
9     def __repr__(self):
10         return "<Atom" + `(` self.feld["Wort"],
11                             self.feld["Tag"],
12                             self.feld["Lemma"] )` + ">"
13
14     def __init__(self, wort, tag, lemma):
15         region.Region.__init__(self)

```



```

16     self.feld = {}
17     self.feld["Wort"] = wort
18     self.feld["Tag"] = tag
19     self.feld["Lemma"] = lemma
20
21     def getFeldinhalt(self, feldname):
22         return [self.feld[feldname]]
23
24     def getDelimiter(self):
25         return "" # muss Wahrheitswert falsch haben
26
27     delimiter = ""
28
29 #}
30
31 class AtomVonZeile(Atom): #{
32
33     def __init__(self, zeile):
34         spalte = string.split(zeile, "\t") # am Tabulator trennen
35         if len(spalte) != 3:
36             self.ok = 0
37             return
38         wort = spalte[0]
39         tag = spalte[1]
40         lemma = spalte[2]
41         lemma = string.rstrip(lemma) # Newline oder anderen WS tilgen
42         Atom.__init__(self, wort, tag, lemma)
43         self.ok = 1
44
45     def isOK(self):
46         return self.ok
47 #}
48
49 if 0: #{
50     a = AtomVonZeile("Äpfel\tNN\tApfel\n")
51     print "Atom a:", a, 'a'
52     b = AtomVonZeile("Birnen\tNN\tBirne\n")
53     print "Atom b:", b, 'b'
54 #}

```

M.1.5 dokument.py

```

1 #!/usr/bin/python
2
3 # Modul Region/dokument
4
5 # KOKS Studienprojekt 2000 - 2001
6 # Korpusbasierte Kollokationensuche
7
8 import absatz
9 import alignment
10 import region
11 import string
12 import sys
13 import time
14
15 class Dokument(region.Regionen): #{
16
17     def __repr__(self):
18         return "<Dokument: " + `self.subregionen` + ">"
19

```

```

20     def getText(self):
21         return string.replace(region.Regionen.getText(self), "\n ", "\n")
22
23     def getDelimiter(self):
24         return "<DOKENDE>\n"
25
26     delimiter = "<DOKENDE>\n"
27
28     def quelleninfo(self):
29         return "Dokument %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)
30
31 #}
32
33 class DokumentVonStream(Dokument): #{
34
35     def __repr__(self):
36         return "<DokumentVonStream>"
37
38     def __init__(self, f): #{
39         zeile = f.readline()
40         absatzliste = []
41         zeilenliste = []
42         ignorieren=0
43         inhaltGelesen = 0
44         while zeile != "": #{
45             if zeile[0:8] == "<ABSATZ>":
46                 absatzliste.append(absatz.AbsatzVonZeilenliste(zeilenliste))
47                 zeilenliste = []
48                 inhaltGelesen = 0
49                 ignorieren=0                # <KOKS> komplett zuruecksetzen
50             elif zeile[0:6] == "<KOKS>":
51                 ignorieren = ignorieren + 1
52             elif zeile[0:7] == "</KOKS>":
53                 ignorieren = ignorieren - 1
54             elif zeile[0:9] == "<DOKENDE>":    # Dokument.delimiter
55                 break
56             elif zeile[0:9] == "<dokende>":    # kommt in den de-news vor
57                 break
58             elif zeile[0:12] == "<KorpusEnde>": # zur Sicherheit
59                 break
60             elif not ignorieren:
61                 zeilenliste.append(zeile)
62                 inhaltGelesen = 1
63                 zeile = f.readline()
64         #}
65         if inhaltGelesen:
66             absatzliste.append(absatz.AbsatzVonZeilenliste(zeilenliste))
67         Dokument.__init__(self, absatzliste)
68     #}
69
70     def quelleninfo(self):
71         return "Dokument von Eingabestrom"
72
73 #}
74
75 class DokumentVonDatei(DokumentVonStream): #{
76
77     def __repr__(self):
78         return "<Dokument: " + self.dateiname + ">"
79
80     def __init__(self, dateiname):

```

```

81     self.dateiname = dateiname
82     f = open(dateiname, "r")
83     DokumentVonStream.__init__(self, f)
84
85     def quelleninfo(self):
86         return "Dokument-Datei %s" %self.dateiname
87
88 #}
89
90 class DokumentPaar(Dokument, alignment.AlignedRegion): #{
91
92     def __init__(self, dok1, dok2, qs, zs):
93         alignment.AlignedRegion.__init__(self, dok1, dok2, qs, zs)
94         Dokument.__init__(self, dok1)
95
96     def alignedRegion(self): # liefert das alignte Dokument
97         return DokumentPaar(self.reg2, self.reg1, self.lang2, self.lang1)
98
99     def getItemAtIndex(self, index):
100         absatz1 = self.reg1[index]
101         try:
102             absatz2 = self.reg2[index]
103         except IndexError:
104             sys.stderr.write("Warnung: %s hat verschieden viele Absaetze\n" %self')
105             return self.reg1[index]
106         retval = absatz.AbsatzPaar(absatz1, absatz2, self.lang1, self.lang2)
107         retval.setQuelleninfo(index, self.quelleninfo())
108         return retval
109
110     def quelleninfo(self):
111         info1 = self.reg1.quelleninfo()
112         info2 = self.reg2.quelleninfo()
113         return "Paar %s und %s" %(info1, info2)
114
115 #}
116
117 class DokumentPaarVonDateien(DokumentPaar): #{
118
119     def __init__(self, fnameqs, fnamezs, qs, zs):
120         dok1 = DokumentVonDatei(fnameqs) # Quellsprache
121         dok2 = DokumentVonDatei(fnamezs) # Zielsprache
122         DokumentPaar.__init__(self, dok1, dok2, qs, zs)
123
124     def quelleninfo(self):
125         info1 = self.reg1.quelleninfo()
126         info2 = self.reg2.quelleninfo()
127         if info1[-7:] == ".de.tag" and info2[-7:] == ".en.tag" \
128             and info1[:-7] == info2[:-7]:
129             return info1[:-7]
130         else:
131             return "Paar %s und %s" %(info1, info2)
132
133 #}
134
135 class DokumentPaarVonStreams(DokumentPaar): #{
136
137     def __init__(self, finqs, finzs, qs, zs):
138         dok1 = DokumentVonStream(finqs) # Quellsprache
139         dok2 = DokumentVonStream(finzs) # Zielsprache
140         DokumentPaar.__init__(self, dok1, dok2, qs, zs)
141

```

```

142 #}
143
144 class DokumentPaarVonRohtexte(DokumentPaar): #[
145
146     def __init__(self, fin1, fin2, qs, zs, verbose = 0):
147         absatzdic = {}
148         absatzdic[qs] = fin1.readline()
149         absatzdic[zs] = fin2.readline()
150         if verbose:
151             sys.stderr.write("tagging %s...\n" % 'absatzdic')
152         getaggt = self.p_taggen(absatzdic)
153         if verbose:
154             for key in getaggt.keys():
155                 sys.stderr.write("result[%s]:\n" % key)
156                 getaggt[key].writeTagzeilen(sys.stderr.write)
157         absatzlistel = []
158         absatzliste2 = []
159         while getaggt:
160             absatzlistel.append(getaggt[qs])
161             absatzliste2.append(getaggt[zs])
162             absatzdic[qs] = fin1.readline()
163             absatzdic[zs] = fin2.readline()
164             if verbose:
165                 sys.stderr.write("tagging %s...\n" % 'absatzdic')
166             getaggt = self.p_taggen(absatzdic)
167             if verbose:
168                 for key in getaggt.keys():
169                     sys.stderr.write("result[%s]:\n" % key)
170                     getaggt[key].writeTagzeilen(sys.stderr.write)
171         dok1 = Dokument(absatzlistel)
172         dok2 = Dokument(absatzliste2)
173         DokumentPaar.__init__(self, dok1, dok2, qs, zs)
174
175     def p_taggen(self, absatzdic):
176         tagdic = {}
177         for key in absatzdic.keys():
178             einAbsatz = absatzdic[key]
179             if not einAbsatz:
180                 return {}
181             getaggt = tagger.tagline(einAbsatz, key)
182             tagdic[key] = getaggt
183         return tagdic
184
185 #}
186
187 import tagger

```

M.1.6 korpus.py

```

1 #!/usr/bin/python
2
3 # Modul Region/korpora
4
5 # KOKS Studienprojekt 2000 - 2001
6 # Korpusbasierte Kollokationensuche
7
8 import alignment
9 import dokument
10 import region
11 import string
12

```

```

13 class Korpus(region.Regionen): #{
14     def __repr__(self):
15         return "<Korpus: " + `self.subregionen`+ ">"
16
17     def getText(self):
18         list = []
19         for i in region.Regionen.keys():
20             list.append(i.getText())
21         return string.join(list, "\n" + 72*"- " + "\n\n")
22
23     def getDelimiter(self):
24         return "<KorpusEnde>\n"
25
26     delimiteri = "<KorpusEnde>\n"
27
28     def quelleninfo(self):
29         return "Korpus\n"
30
31 #}
32
33 class KorpusMitSprache(Korpus): #{
34
35     def __init__(self, dokumentliste, sprache):
36         self.sprache = sprache
37         Korpus.__init__(self, dokumentliste)
38
39     def p_setSprache(self, sprache):    # nur in dieser Klasse
40         self.sprache = sprache
41
42     def p_setDokumentliste(self, liste):    # nur in dieser Klasse
43         self.subregionen = liste
44
45 #}
46
47 class KorpusVonDateiliste(Korpus): #{
48
49     def __init__(self, dateiliste):
50         liste = []
51         for i in dateiliste:
52             liste.append(dokument.DokumentVonDatei(i))
53         Korpus.__init__(self, liste)
54 #}
55
56 class KorpusPaar(Korpus, alignment.AlignedRegion): #{
57
58     def __init__(self, korp1, korp2, lang1, lang2):
59         alignment.AlignedRegion.__init__(self, korp1, korp2, lang1, lang2)
60         Korpus.__init__(self, korp1)    # Quellsprache
61
62     def alignedRegion(self):    # liefert den alignnten Korpus
63         return KorpusPaar(self.reg2, self.reg1, self.lang2, self.lang1)
64
65     def getItemAtIndex(self, index):
66         dok1 = self.reg1[index]
67         try:
68             dok2 = self.reg2[index]
69         except IndexError:
70             return self.reg1[index]
71         return dokument.DokumentPaar(dok1, dok2, self.lang1, self.lang2)
72
73 #}

```

M.1.7 region.py

```

1  #!/usr/bin/python
2
3  # abstrakte Basisklasse fuer alle Arten von Regionen: Atom, Segment,
4  # Absatz und Dokument.
5
6  import bisect
7  import string
8  import sys
9  import types
10
11 class Region: #{
12
13     def __init__(self):
14         self.setQuelleninfo(-1, "unbekannt")
15         #self.cache = cache.DefaultCache(self)
16
17     def __repr__(self):
18         return "<abstrakte Region>"           # diese Methoden ueberschreiben
19
20     def p_getFeldinhalt(self, feldname):
21         if self.cache.has_key(feldname):
22             return self.cache[feldname]
23         retval = self.getFeldinhalt(feldname)
24         self.cache[feldname] = retval
25         return retval
26
27     def getFeldinhalt(self, feldname):         # diese Methoden ueberschreiben
28         return []
29
30     def getWoerter(self):
31         return self.getFeldinhalt("Wort")
32
33     def getTags(self):
34         return self.getFeldinhalt("Tag")
35
36     def getLemmata(self):
37         return self.getFeldinhalt("Lemma")
38
39     def getText(self):
40         return string.join(self.getWoerter())
41
42     def __str__(self):
43         return self.getText()
44
45     # def getDelimiter(self):                 # Interface !!!
46     #     return "<Delimiter>\n"
47     #
48     #     delimiter = "<Delimiter>\n"
49
50     def writeTagzeilen(self, fun):
51         mm=map(None, self.getWoerter(),
52                self.getTags(),
53                self.getLemmata())
54         for (w,t,l) in mm:
55             apply(fun, ("%s\t%s\t%s\n" %(w,t,l),))
56
57     def quelleninfo(self):
58         return "Region %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)
59

```

```

60     def setQuelleninfo(self, index, von):
61         self.quelleninfo_index = index
62         self.quelleninfo_von = von
63
64     #}
65
66     class Regionen(Region): #{
67
68         def __repr__(self):                # diese Methoden ueberschreiben
69             return "<abstrakte Regionen>"
70
71         def __init__(self, regionenliste):
72             Region.__init__(self)
73             self.subregionen = regionenliste
74
75         def getFeldinhalt(self, feldname):
76             liste = []
77             for i in self.keys():
78                 liste = liste + self[i].getFeldinhalt(feldname)
79             return liste
80
81         def getText(self):                # nicht von Region uebernommen,
82             liste = []                    # damit Aenderungen an getText
83             for i in self.keys():         # auch weitergereicht werden
84                 liste.append(self[i].getText())
85             return string.join(liste)
86
87         def __len__(self):                # ueberschreiben, wenn getRegionenZu
88             return len(self.subregionen) # funktionieren soll
89
90         def has_key(self, key):
91             return 0 <= key < len(self)
92
93         def keys(self):
94             return range(0, len(self))
95
96         def getItemAtIndex(self, index):
97             retval = self.subregionen[index]
98             retval.setQuelleninfo(index, self.quelleninfo())
99             return retval
100
101         def index(self, obj):
102             for i in range(0, self.__len__()):
103                 cmp = self[i]
104                 if cmp is obj:
105                     return i
106             raise ValueError, "%s not in list!" %obj
107
108         def count(self, obj):
109             counter = 0
110             for i in range(0, self.__len__()):
111                 cmp = self[i]
112                 #print cmp, obj
113                 if cmp is obj:
114                     counter = counter + 1
115             return counter
116
117         def __getitem__(self, key):
118             if type(key) == types.IntType:
119                 return self.getItemAtIndex(key)
120             if type(key) == types.SliceType:

```

```

121         return RegionenSlice(self, key)
122     raise TypeError
123
124     def __getslice__(self, a, b):          # Python 1.5 Kompatibilitaet
125         return RegionenSlice(self, Slice15(a, b))
126
127     def getRegionenMit(self, klasse):
128         return RegionenMit(self, klasse)
129
130     def writeTagzeilen(self, fun):
131     #         apply(fun, ("DEBUG: vor keys() %s" %self.keys(),))
132         for i in self.keys():              # auch weitergereicht werden
133             self[i].writeTagzeilen(fun)
134     #         apply(fun, ("DEBUG: nach schleife",))
135         try:
136             delimiter = self.getDelimiter()
137         except AttributeError:
138             delimiter = "<unbestimmter Regionentyp>\n"
139         apply(fun, ("%s" %delimiter,))
140
141     #}
142
143     class Slice15: #{                          # Hilfsklasse zur Python 1.5 Kompatibilitaet
144         def __init__(self, a, b):
145             self.start = a
146             self.stop   = b
147             self.step   = None
148     #}
149
150     class RegionenMit(Regionen): #{
151
152         def __repr__(self):
153             return "<Regionen mit " + self.klasse.__name__ + ":%s>" %self.p_repr2()
154
155         def p_repr2(self):
156             return "\n\t"+`(self.pos, self.subregionen)`
157
158         def __init__(self, regionen, klasse): #{
159             self.klasse = klasse
160             delimiter = klasse.delimiter          # workaround: delimiter benutzen
161             if len(regionen) == 0:
162                 self.pos = [0]
163                 Regionen.__init__(self, [])
164             #elif isinstance(regionen[0], klasse):
165             elif regionen[0].getDelimiter() == delimiter:
166                 self.pos = [0, len(regionen)]
167                 Regionen.__init__(self, [regionen])
168             else: #{
169                 liste = []
170                 start = 0
171                 self.pos = [0]
172                 import time
173                 import sys
174                 letzteAusgabe = 0
175                 rname = regionen.__class__.__name__
176                 kname = klasse.__name__
177                 for i in regionen.keys(): #{
178                     subregionenMit = RegionenMit(regionen[i], klasse)
179                     liste.append(subregionenMit)
180                     start = start + len(subregionenMit)
181                     self.pos.append(start)

```



```

182         #if time.time() - letzteAusgabe >= 1.0:
183         #     sys.stderr.write('start'+ " "+kname+"-Regionen zu " + rname + "\r")
184         #     letzteAusgabe = time.time()
185     #}
186     #sys.stderr.write('start'+ " "+kname+"-Regionen zu " + rname + ".\n")
187     Regionen.__init__(self, liste)
188 #}
189 #}
190
191 def __len__(self):
192     retval = self.pos[len(self.pos)-1]
193     if retval < 0:
194         sys.stderr.write("RegionenMit: len < 0\n")
195     return retval
196
197 def getItemAtIndex(self, index):
198     key = index
199     pos = bisect.bisect(self.pos, key)-1
200     try:
201         return self.subregionen[pos][key - self.pos[pos]]
202     except:
203         raise IndexError
204
205 #}
206
207
208 class RegionenSlice(Regionen): #{
209
210     def __repr__(self):
211         return "<" + `self.reg` + "[" + `self.start` + \
212             ":" + `self.stop` + \
213             ", " + `self.step` + "]">"
214
215     def __init__(self, reg, slice):
216         self.reg = reg
217         start, stop, step = (slice.start, slice.stop, slice.step)
218         if not start:
219             start = 0
220         if not stop:
221             stop = len(reg)
222         if start < 0:
223             start = start + len(reg)
224         if start < 0:
225             start = 0
226         if stop < 0:
227             stop = stop + len(reg)
228         if stop < 0:
229             stop = 0
230         if stop > len(reg):
231             stop = len(reg)
232         if start > len(reg):
233             start = len(reg)
234         if start > stop:
235             start = stop
236         self.start = start
237         self.stop = stop
238         self.step = step
239         if self.start != slice.start or self.stop != self.stop:
240             sys.stderr.write("Warnung: Slice %s -> %s, da len=%s\n" %('slice', `self`, `len(reg)'))
241         if step:
242             sys.stderr.write('self' + " noch nicht implementiert\n")

```

```

243
244     def __getstate__(self):
245         return (self.reg, self.start, self.stop, self.step)
246
247     def __setstate__(self, state):
248         self.reg, self.start, self.stop, self.step = state
249
250     def __len__(self):
251         retval = self.stop - self.start
252         if retval < 0:
253             sys.stderr.write("RegionenSlice: len < 0\n")
254         return retval
255
256     def getItemAtIndex(self, index):
257         key = index + self.start
258         if key >= self.stop:
259             raise IndexError
260         return self.reg[key]
261
262     def getDelimiter(self):
263         return self.reg.getDelimiter()
264
265 #}
266
267 from DatabaseAPI import cache

```

M.1.8 satz.py

```

1  #!/usr/bin/python
2
3  # Modul Region.satz
4
5  import atom
6  import region
7
8  class Satz(region.Regionen): #{
9
10     def __repr__(self):
11         return "<Satz: " + `self.subregionen`+ ">"
12
13     def getDelimiter(self):
14         return "<SATZ>\n"
15
16     delimiter = "<SATZ>\n"
17
18 #}
19
20 class SatzVonZeilenliste(Satz): #{
21     def __init__(self, zeilenliste):
22         atomliste = []
23         for zeile in zeilenliste:
24             a = atom.AtomVonZeile(zeile)
25             if a.isOK():
26                 atomliste.append(a)
27         Satz.__init__(self, atomliste)
28 #}
29
30 if 0: #{
31     a = "Äpfel\tNN\tApfel\n"
32     b = "Birnen\tNN\tBirne\n"
33     c = SatzVonZeilenliste([a,b])

```

```

34 print "Satz [a,b]:", c, 'c`
35 r = c.getRegionenMit(atom.Atom)
36 print r, `r`, len(r)
37 for i in range(0, len(r)):
38     print ("["+i`+"]: " + `r[i]`)
39 print "Lemmata:", r.getLemmata()
40 #}

```

M.1.9 segment.py

```

1  #!/usr/bin/python
2
3  # Modul Region.segment
4
5  # KOKS Studienprojekt 2000 - 2001
6  # Korpusbasierte Kollokationensuche
7
8  import alignment
9  import atom
10 import region
11 import satz
12 import string
13
14 class Segment(region.Regionen): #{
15
16     def __repr__(self):
17         return "<Segment: " + `self.subregionen`+ ">"
18
19     def getDelimiter(self):
20         return "<segmentgrenze>\n"
21
22     delimiter = "<segmentgrenze>\n"
23
24     def quelleninfo(self):
25         return "Segment %s von %s" %(self.quelleninfo_index, self.quelleninfo_von)
26
27 #}
28
29
30 class SegmentPaar(Segment, alignment.AlignedRegion): #{
31
32     def __init__(self, seg1, seg2, qs, zs):
33         alignment.AlignedRegion.__init__(self, seg1, seg2, qs, zs)
34         Segment.__init__(self, seg1)
35
36     def alignedRegion(self): # liefert das alignte Segment
37         return SegmentPaar(self.reg2, self.reg1, self.lang2, self.lang1)
38
39 #}
40
41
42 class SegmentVonZeilenliste(Segment): #{
43
44     def __repr__(self):
45         return "<Segment: " + string.strip(self.getText()[0:80]) + ">"
46
47     def __init__(self, zeilenliste):
48         satzliste = []
49         pos = 0
50         start = 0
51         inhaltGelesen = 0

```

```

52     for zeile in zeilenliste: #{
53         if zeile[0:6] == "<SATZ>": #{
54             satzliste.append(
55                 satz.SatzVonZeilenliste(zeilenliste[start:pos]))
56             start = pos + 1
57             inhaltGelesen = 0
58         #}
59         else: #{
60             inhaltGelesen = 1
61         #}
62         pos = pos + 1
63     #}
64     if inhaltGelesen: #{           # Inhalt nach dem letzten Delimiter?
65         satzliste.append(
66             satz.SatzVonZeilenliste(zeilenliste[start:pos]))
67     #}
68     Segment.__init__(self, satzliste)
69 #}
70
71 if 0: #{
72     a = "Äpfel\tNN\tApfel\n"
73     b = "Birnen\tNN\tBirne\n"
74     c = SegmentVonZeilenliste([a,b])
75     print "Segment [a,b]:", c, 'c'
76     r = c.getRegionenMit(atom.Atom)
77     print r, 'r', len(r)
78     for i in range(0, len(r)):
79         print ("["+i+"]: " + 'r[i]')
80     print "Lemmata:", r.getLemmata()
81 #}

```

M.2 Church and Gale Aligner

Den Aligner von Church und Gale haben wir an einigen Stellen verändert.

M.2.1 align.c

```

1  /*
2  The following code is the core of align.
3  It is a C language program which inputs two text files, with one
4  token (word) per line. The text files contain a number of delimiter tokens.
5  There are two types of delimiter tokens: ``hard`` and ``soft``.
6  The hard regions (e.g., paragraphs) may not be changed, and there must be
7  equal numbers of them in the two input files.
8  The soft regions (e.g., sentences) may be deleted (1-0), inserted (0-1),
9  substituted (1-1), contracted (2-1), expanded (1-2), or merged (2-2)
10 as necessary so that the output ends up with the same number of soft regions.
11 The program generates two output files. The two output files contain an
12 equal number of soft regions, each on a line. If the -v command line option is
13 included, each soft region is preceded by its probability score.
14 */
15
16 #include <malloc.h>
17 #include <math.h>
18 #include <stdio.h>
19 #include <string.h>
20 #include <stdlib.h>
21 #ifdef POSIX
22 # include <unistd.h>
23 #else

```

```

24 # define HAVE_STRING_H 1
25 # include "getopt.h"
26 #endif
27
28 #define MAXINT (~0 > 0 ? ~0 : (unsigned) ~0 >> 1)
29
30 /*
31  usage:
32  align_regions -D '.PARA' -d '.End of Sentence' file1 file2
33
34  outputs two files: file1.al & file2.al
35
36  hard regions are delimited by the -D arg
37  soft regions are delimited by the -d arg
38 */
39
40 static int endRegionsWithDelimiter = 0;
41
42 #define dist(x,y) distances[(x) * ((ny) + 1) + (y)]
43 #define pathx(x,y) path_x[(x) * ((ny) + 1) + (y)]
44 #define pathy(x,y) path_y[(x) * ((ny) + 1) + (y)]
45 #define MAX_FILENAME 256
46 #define BIG_DISTANCE 2500
47
48 /* Dynamic Programming Optimization */
49 struct alignment {
50     int x1;
51     int y1;
52     int x2;
53     int y2;
54     int d;
55 };
56
57 char *hard_delimiter = NULL; /* -D arg */
58 char *soft_delimiter = NULL; /* -d arg */
59 int verbose = 0; /* -v arg */
60
61 /* utility functions */
62 char *readchars(char *filename, int *len_ptr),
63     **readlines(char *filename, int *len_ptr),
64     **substrings(char *string, char *end, char delimiter, int *len_ptr);
65 void err(char *msg);
66
67 /*
68  seq_align by Mike Riley
69
70  x and y are sequences of objects, represented as non-zero ints, to be aligned.
71
72  dist_func(x1, y1, x2, y2) is a distance function of 4 args:
73
74  dist_func(x1, y1, 0, 0) gives cost of substitution of x1 by y1.
75  dist_func(x1, 0, 0, 0) gives cost of deletion of x1.
76  dist_func(0, y1, 0, 0) gives cost of insertion of y1.
77  dist_func(x1, y1, x2, 0) gives cost of contraction of (x1,x2) to y1.
78  dist_func(x1, y1, 0, y2) gives cost of expansion of x1 to (y1,y2).
79  dist_func(x1, y1, x2, y2) gives cost to match (x1,x2) to (y1,y2).
80
81  align is the alignment, with (align[i].x1, align[i].x2) aligned
82  with (align[i].y1, align[i].y2). Zero in align[].x1 and align[].y1
83  correspond to insertion and deletion, respectively. Non-zero in
84  align[].x2 and align[].y2 correspond to contraction and expansion,

```

```

85     respectively. align[].d gives the distance for that pairing.
86
87     The function returns the length of the alignment.
88 */
89
90 int
91 seq_align(
92     int *x, int *y, int nx, int ny,
93     int (*dist_func)(),
94     struct alignment **align)
95 {
96     int *distances, *path_x, *path_y, n;
97     int i, j, oi, oj, di, dj, d1, d2, d3, d4, d5, d6, dmin;
98     struct alignment *ralign;
99     distances = (int *) malloc((nx + 1) * (ny + 1) * sizeof(int));
100    path_x = (int *) malloc((nx + 1) * (ny + 1) * sizeof(int));
101    path_y = (int *) malloc((nx + 1) * (ny + 1) * sizeof(int));
102    ralign = (struct alignment *) malloc((nx + ny) * sizeof(struct alignment));
103    for(j = 0; j <= ny; j++) {
104        for(i = 0; i <= nx; i++) {
105            d1 = i>0 && j>0 ? /* substitution */
106                dist(i-1, j-1) + (*dist_func)(x[i-1], y[j-1], 0, 0)
107                : MAXINT;
108            d2 = i>0 ? /* deletion */
109                dist(i-1, j) + (*dist_func)(x[i-1], 0, 0, 0)
110                : MAXINT;
111            d3 = j>0 ? /* insertion */
112                dist(i, j-1) + (*dist_func)(0, y[j-1], 0, 0)
113                : MAXINT;
114            d4 = i>1 && j>0 ? /* contraction */
115                dist(i-2, j-1) + (*dist_func)(x[i-2], y[j-1], x[i-1], 0)
116                : MAXINT;
117            d5 = i>0 && j>1 ? /* expansion */
118                dist(i-1, j-2) + (*dist_func)(x[i-1], y[j-2], 0, y[j-1])
119                : MAXINT;
120            d6 = i>1 && j>1 ? /* melding */
121                dist(i-2, j-2) + (*dist_func)(x[i-2], y[j-2], x[i-1], y[j-1])
122                : MAXINT;
123
124            dmin = d1;
125            if(d2<dmin) dmin=d2;
126            if(d3<dmin) dmin=d3;
127            if(d4<dmin) dmin=d4;
128            if(d5<dmin) dmin=d5;
129            if(d6<dmin) dmin=d6;
130
131            if(dmin == MAXINT) {
132                dist(i,j) = 0;
133            }
134            else if(dmin == d1) {
135                dist(i,j) = d1;
136                pathx(i,j) = i-1;
137                pathy(i,j) = j-1;
138            }
139            else if(dmin == d2) {
140                dist(i,j) = d2;
141                pathx(i,j) = i-1;
142                pathy(i,j) = j;
143            }
144            else if(dmin == d3) {
145                dist(i,j) = d3;

```

```

146     pathx(i,j) = i;
147     pathy(i,j) = j-1;
148 }
149 else if(dmin == d4) {
150     dist(i,j) = d4;
151     pathx(i,j) = i-2;
152     pathy(i,j) = j-1;
153 }
154 else if(dmin == d5){
155     dist(i,j) = d5;
156     pathx(i,j) = i-1;
157     pathy(i,j) = j-2;
158 }
159 else /* dmin == d6 */ {
160     dist(i,j) = d6;
161     pathx(i,j) = i-2;
162     pathy(i,j) = j-2;
163 }
164 }
165 }
166 n = 0;
167 for(i=nx, j=ny ; i>0 || j>0 ; i = oi, j = oj) {
168     oi = pathx(i, j);
169     oj = pathy(i, j);
170     di = i - oi;
171     dj = j - oj;
172
173     if(di == 1 && dj == 1) { /* substitution */
174         ralign[n].x1 = x[i-1];
175         ralign[n].y1 = y[j-1];
176         ralign[n].x2 = 0;
177         ralign[n].y2 = 0;
178         ralign[n++].d = dist(i, j) - dist(i-1, j-1);
179     }
180
181     else if(di == 1 && dj == 0) { /* deletion */
182         ralign[n].x1 = x[i-1];
183         ralign[n].y1 = 0;
184         ralign[n].x2 = 0;
185         ralign[n].y2 = 0;
186         ralign[n++].d = dist(i, j) - dist(i-1, j);
187     }
188
189     else if(di == 0 && dj == 1) { /* insertion */
190         ralign[n].x1 = 0;
191         ralign[n].y1 = y[j-1];
192         ralign[n].x2 = 0;
193         ralign[n].y2 = 0;
194         ralign[n++].d = dist(i, j) - dist(i, j-1);
195     }
196
197     else if(dj == 1) { /* contraction */
198         ralign[n].x1 = x[i-2];
199         ralign[n].y1 = y[j-1];
200         ralign[n].x2 = x[i-1];
201         ralign[n].y2 = 0;
202         ralign[n++].d = dist(i, j) - dist(i-2, j-1);
203     }
204
205     else if(di == 1) { /* expansion */
206         ralign[n].x1 = x[i-1];

```

```

207         ralign[n].y1 = y[j-2];
208         ralign[n].x2 = 0;
209         ralign[n].y2 = y[j-1];
210         ralign[n++].d = dist(i, j) - dist(i-1, j-2);
211     }
212
213     else /* di == 2 && dj == 2 */ { /* melding */
214         ralign[n].x1 = x[i-2];
215         ralign[n].y1 = y[j-2];
216         ralign[n].x2 = x[i-1];
217         ralign[n].y2 = y[j-1];
218         ralign[n++].d = dist(i, j) - dist(i-2, j-2);
219     }
220 }
221 *align = (struct alignment *) malloc(n * sizeof(struct alignment));
222 for(i=0; i<n; i++)
223     memcpy((*align) + (n-i-1), ralign + i, sizeof(struct alignment));
224 free(distances);
225 free(path_x);
226 free(path_y);
227 free(ralign);
228 return(n);
229 }
230
231 /* Local Distance Function */
232 /* Returns the area under a normal distribution
233    from -inf to z standard deviations */
234 double
235 pnorm(
236     double z)
237 {
238     double t, pd;
239     t = 1/(1 + 0.2316419 * z);
240     pd = 1 - 0.3989423 *
241         exp(-z * z/2) *
242         (((1.330274429 * t - 1.821255978) * t
243           + 1.781477937) * t - 0.356563782) * t + 0.319381530) * t;
244     /* see Abramowitz, M., and I. Stegun (1964), 26.2.17 p. 932 */
245     return(pd);
246 }
247
248 /* Return -100 * log probability that an English sentence of length
249    len1 is a translation of a foreign sentence of length len2. The
250    probability is based on two parameters, the mean and variance of
251    number of foreign characters per English character.
252 */
253
254 int
255 match(
256     int len1, int len2)
257 {
258     double z, pd, mean;
259     double c = 1;
260     double s2 = 6.8 ;
261
262     if(len1==0 && len2==0) return(0);
263     mean = (len1 + len2/c)/2;
264     z = (c * len1 - len2)/sqrt(s2 * mean);
265
266     /* Need to deal with both sides of the normal distribution */
267     if(z < 0) z = -z;

```



```

268     pd = 2 * (1 - pnorm(z));
269
270     if(pd > 0) return((int)(-100 * log(pd)));
271     else return(BIG_DISTANCE);
272 }
273
274 int
275 two_side_distance(
276     int x1, int y1, int x2, int y2)
277 {
278     int penalty21 = 230;          /* -100 * log([prob of 2-1 match] / [prob of 1-1 match]) */
279     int penalty22 = 440;          /* -100 * log([prob of 2-2 match] / [prob of 1-1 match]) */
280     int penalty01 = 450; /* -100 * log([prob of 0-1 match] / [prob of 1-1 match]) */
281
282     if(x2 == 0 && y2 == 0)
283
284         if(x1 == 0) /* insertion */
285             return(match(x1, y1) + penalty01);
286
287         else if(y1 == 0) /* deletion */
288             return(match(x1, y1) + penalty01);
289
290
291         else return (match(x1, y1)); /* substitution */
292
293     else if(x2 == 0) /* expansion */
294         return (match(x1, y1 + y2) + penalty21);
295
296     else if(y2 == 0) /* contraction */
297         return(match(x1 + x2, y1) + penalty21);
298
299     else /* merger */
300         return(match(x1 + x2, y1 + y2) + penalty22);
301 }
302
303 /* Functions for Manipulating Regions */
304
305 struct region {
306     char **lines;
307     int length;          /* number of lines in region */
308 };
309
310 void
311 print_region(
312     FILE *fd,
313     struct region *region,
314     int score)
315 {
316     char **lines, **end;
317
318     lines = region->lines;
319     end = lines + region->length; /* points one entry behind array */
320     for( ; lines < end ; lines++) /* print out all the valid lines */
321         fprintf(fd, "%s\n", *lines);
322 }
323
324 /*
325  * length_of_a_region(region)
326  * returns amount of memory needed for lines of region region
327  * serves as a distance-measure for seq_align()
328  */

```

```

329
330 int
331 length_of_a_region(
332     struct region *region)
333 {
334     int result;
335     char **lines, **end;
336
337     lines = region->lines;
338     end = lines + region->length;
339     result = end - lines;          /* number of lines in region */
340
341     for( ; lines < end; lines++) /* add total chars in region */
342         result += strlen(*lines);
343     return(result);
344 }
345
346 int *
347 region_lengths(
348     struct region *regions,
349     int n)
350 {
351
352     int i;
353     int *result;
354
355     result = (int *)malloc(n * sizeof(int));
356     if(result == NULL) err("malloc failed");
357
358     for(i = 0; i < n; i++)
359         result[i] = length_of_a_region(regions+i);
360     return(result);
361 }
362
363 struct region *
364 find_sub_regions(
365     struct region *region,
366     char *delimiter,
367     int *len_ptr)
368 {
369     struct region *result;
370     char **l, **lines, **end;
371     int n = 0;
372
373     lines = region->lines;
374     end = lines + region->length;
375
376     for(l = lines; l < end; l++)
377         if(delimiter && strcmp(*l, delimiter) == 0) n++;
378
379     if (verbose) {
380         if (delimiter)
381             fprintf(stderr, "Number of delimiters %s found in region %p: %d\n",
382                 delimiter, region, n);
383         else
384             fprintf(stderr, "NULL delimiter\n");
385     }
386
387     result = (struct region *)calloc(n+2, sizeof(struct region)); /* +1 should also work */
388     if(result == NULL) err("malloc failed");
389     *len_ptr = n;

```

```

390 if (!endRegionsWithDelimiter)
391     (*len_ptr)++;
392 n = 0;
393 result[0].lines = lines;
394 for(l = lines; l < end; l++)
395     if(delimiter && strcmp(*l, delimiter) == 0) {
396         result[n].length = l - result[n].lines;
397         result[n+1].lines = l+1;
398         n++;
399     }
400 result[n].length = l - result[n].lines;
401 if (!endRegionsWithDelimiter)
402     n++;
403 if(n != *len_ptr) {
404     fprintf(stderr, "find_sub_regions: n = %d, *len_ptr = %d\n", n, *len_ptr);
405     exit(2);
406 }
407 if (verbose) {
408     fprintf(stderr, "returning %d regions: %p upto %p\n",
409             *len_ptr, result, result+(*len_ptr)-1);
410 }
411 return(result);
412 }
413
414 /* Top Level Main Function */
415 int
416 main(
417     int argc,
418     char **argv)
419 {
420     char **lines1, **lines2;
421     int number_of_lines1, number_of_lines2;
422     struct region *hard_regions1, *hard_regions2, *soft_regions1, *soft_regions2;
423     struct region *hard_end1, *hard_end2, tmp;
424     int number_of_hard_regions1;
425     int number_of_hard_regions2;
426     int number_of_soft_regions1;
427     int number_of_soft_regions2;
428     int *len1, *len2;
429     int c, n, i, ix, iy, prevx, prevy;
430     struct alignment *align, *a;
431     FILE *out1, *out2;
432     char filename[MAX_FILENAME];
433     extern char *optarg;
434     extern int optind;
435
436     /* parse arguments */
437     while((c = getopt(argc, argv, "ved:D:")) != EOF)
438         switch(c) {
439             case 'v':
440                 verbose = 1;
441                 break;
442             case 'e':
443                 endRegionsWithDelimiter = 1;    /* force original behaviour */
444                 break;
445             case 'd':
446                 soft_delimiter = strdup(optarg);
447                 break;
448             case 'D':
449                 hard_delimiter = strdup(optarg);
450                 break;

```

```

451     default:
452         fprintf(stderr, "usage: align_regions [d (soft delimiter)] [D (hard delimiter)]\n");
453         exit(2);
454     }
455
456     if(argc != optind + 2) err("wrong number of arguments");
457
458     /* open output files */
459     sprintf(filename, "%s.al", argv[optind]);
460     out1 = fopen(filename, "w");
461     if(out1 == NULL) {
462         fprintf(stderr, "can't open %s\n", filename);
463         exit(2);
464     }
465
466     sprintf(filename, "%s.al", argv[optind+1]);
467     out2 = fopen(filename, "w");
468     if(out2 == NULL) {
469         fprintf(stderr, "can't open %s\n", filename);
470         exit(2);
471     }
472
473     lines1 = readlines(argv[optind], &number_of_lines1);
474     lines2 = readlines(argv[optind+1], &number_of_lines2);
475
476     if(verbose)
477         fprintf(stderr, "Processing input file 1.\n");
478
479     tmp.lines = lines1;
480     tmp.length = number_of_lines1;
481     hard_regions1 = find_sub_regions(&tmp, hard_delimiter, &number_of_hard_regions1);
482
483     if(verbose)
484         fprintf(stderr, "Processing input file 2.\n");
485
486     tmp.lines = lines2;
487     tmp.length = number_of_lines2;
488     hard_regions2 = find_sub_regions(&tmp, hard_delimiter, &number_of_hard_regions2);
489
490     if(verbose) {
491         fprintf(out1, "Hard regions: %d\n", number_of_hard_regions1);
492         fprintf(out2, "Hard regions: %d\n", number_of_hard_regions2);
493     }
494
495     if(number_of_hard_regions1 != number_of_hard_regions2) {
496         fprintf(stderr, "Hard regions in file 1: %d\n", number_of_hard_regions1);
497         fprintf(stderr, "Hard regions in file 2: %d\n", number_of_hard_regions2);
498         err("align_regions: input files do not contain the same number of hard regions");
499     }
500
501     if(verbose)
502         fprintf(stderr, "Number of hard regions OK.\n");
503
504     hard_end1 = hard_regions1 + number_of_hard_regions1;
505     hard_end2 = hard_regions2 + number_of_hard_regions2;
506
507     for( ; hard_regions1 < hard_end1 ; hard_regions1++, hard_regions2++) {
508
509         soft_regions1 = find_sub_regions(hard_regions1+0, soft_delimiter,
510             &number_of_soft_regions1);
511         soft_regions2 = find_sub_regions(hard_regions2+0, soft_delimiter,

```

```

512     &number_of_soft_regions2);
513
514     len1 = region_lengths(soft_regions1, number_of_soft_regions1);
515     len2 = region_lengths(soft_regions2, number_of_soft_regions2);
516
517     n = seq_align(len1, len2, number_of_soft_regions1, number_of_soft_regions2,
518                 two_side_distance, &align);
519
520     prevx = prevy = ix = iy = 0;
521     for(i = 0; i < n; i++) {
522         a = &align[i];
523         if(a->x2 > 0) ix++; else if(a->x1 == 0) ix--;
524         if(a->y2 > 0) iy++; else if(a->y1 == 0) iy--;
525         if(a->x1 == 0 && a->y1 == 0 && a->x2 == 0 && a->y2 == 0) {ix++; iy++;}
526         ix++;
527         iy++;
528         if(verbose) {
529             fprintf(out1, ".Score %d\n", a->d);
530             fprintf(out2, ".Score %d\n", a->d);
531         }
532
533         for( ; prevx < ix; prevx++)
534             print_region(out1, soft_regions1+prevx, a->d);
535         if (((i+1) < n) || endRegionsWithDelimiter)
536             fprintf(out1, "%s\n", soft_delimiter);
537
538         for( ; prevy < iy; prevy++)
539             print_region(out2, soft_regions2+prevy, a->d);
540         if (((i+1) < n) || endRegionsWithDelimiter)
541             fprintf(out2, "%s\n", soft_delimiter);
542     }
543     if (((hard_regions1+1) < hard_end1) || endRegionsWithDelimiter) {
544         fprintf(out1, "%s\n", hard_delimiter);
545         fprintf(out2, "%s\n", hard_delimiter);
546     }
547     free(align);
548     free(soft_regions1);
549     free(soft_regions2);
550     free(len1);
551     free(len2);
552 }
553
554 return 0;
555 }
556
557 /* Utility Functions */
558
559 void
560 err(
561     char *msg)
562 {
563     fprintf(stderr, "***ERROR** %s\n", msg);
564     exit(2);
565 }
566
567 /* return the size of a file */
568 long filesize(
569     FILE *fd)
570 {
571     long curpos, length;
572     curpos = ftell(fd);          /* save the current location in the file */

```

```

573     fseek(fd, 0L, SEEK_END);      /* seek to the end of the file */
574     length = ftell(fd);          /* get the current offset into the file */
575     fseek(fd, curpos, SEEK_SET); /* restore saved position */
576     return length;
577 }
578
579 /* return the contents of the file as a string
580    and stuff the length of this string into len_ptr */
581 char *
582 readchars(
583     char *filename,
584     int *len_ptr)
585 {
586     FILE *fd;
587     char *result;
588
589     fd = fopen(filename, "r");
590     if(fd == NULL) err("open failed");
591
592     *len_ptr = filesize(fd);
593
594     result = malloc(*len_ptr);
595     if(result == NULL) err("malloc failed\n");
596
597     if(fread(result, sizeof(char), *len_ptr, fd) != *len_ptr)
598         err("fread failed");
599
600     if(fclose(fd) == -1)
601         err("fclose failed");
602
603     return(result);
604 }
605
606 /* split string into a number of substrings delimited by a delimiter character
607    return an array of substrings
608    stuff the length of this array into len_ptr
609    attention: leading delimiter characters are skipped
610 */
611 char **
612 substrings(
613     char *string, char *end, char delimiter,
614     int *len_ptr)
615 {
616     char *s, **result;
617     int i = 0;
618
619     while(string < end && *string == delimiter) string++;
620
621     for(s = string; s < end; s++)
622         if(*s == delimiter) i++;
623     *len_ptr = i;
624
625     result = (char **)malloc(sizeof(char *) * (i+1));
626     if(result == NULL) err("malloc failed");
627
628     i = 0;
629     result[i++] = string;
630     for(s = string; s < end; s++)
631         if(*s == delimiter) {
632             result[i++] = s+1;
633             *s = 0;

```

```

634     }
635     i--;          /*the last entry is beyond the end*/
636     if(i != *len_ptr) {
637         fprintf(stderr, "align_regions: confusion; i= %d; *len_ptr = %d\n", i, *len_ptr);
638         exit(2);
639     }
640
641     return(result);
642 }
643
644 /* return an array of strings, one string for each line of the file
645    set len_ptr to the number of lines in the file */
646 char **
647 readlines(
648     char *filename,
649     int *len_ptr)
650 {
651     char *chars;
652     int number_of_chars;
653     chars = readchars(filename, &number_of_chars);
654     return(substrings(chars, chars + number_of_chars, '\n', len_ptr));
655 }

```

M.2.2 getopt.c und getopt.h

Für nicht POSIX Systeme ist align.c bereits darauf vorbereitet, dass ggf. die getopt Quellen eingebunden werden müssen. Den Quellcode des Moduls getopt kann der GNU C-Library glibc entnommen werden. Auf diese Weise wurde align.c mit dem Watcom C/C++ Compiler 10.6 unter MS Windows 95 mit getopt.[ch] aus der glibc 2.2.0 übersetzt. Das Makefile (siehe M.4.1) geht von einem POSIX System aus.

M.3 Weitere Python Module und Skripte

...

M.3.1 abstand.py

Siehe auch Beschreibung der Abstandsmaße ab Seite 55 und insbesondere Seite 62 wegen eines Fehlers im TG2 Abstandsmaß.

```

1  #!/usr/bin/python
2
3  # Modul abstand
4
5  import zeitmessung
6
7  import math
8  import popen2
9  import string
10 import StringIO
11 import sys
12
13 class Abstand: #{
14
15     def __init__(self, default = 0.0):          # aufrufen wg. aiout
16         self.default = default
17         self.aiout=NoneFile()
18
19     def setAiout(self, out):
20         self.aiout = out
21

```

```

22     def __repr__(self):                # diese Methoden ueberschreiben
23         return "<Abstand immer %s>" %self.default
24
25     def zwischen(self, reg1, reg2):    # diese Methode ueberschreiben
26         self.aiout.write("Dummyabstand %s\n" %self.default)
27         self.aiout.write("<SATZ>\n")
28         return self.default
29
30     def __call__(self, reg1, reg2):    # bequemer Aufruf
31         return self.zwischen(reg1, reg2)
32
33     def statistic(self):
34         return "keine Statistik vorhanden"
35
36     def close(self):
37         pass
38
39 #}
40
41 class AbstandEQU(Abstand): #{
42
43     def __init__(self):
44         Abstand.__init__(self)
45
46     def __repr__(self):
47         return "<AbstandEQU>"
48
49     def zwischen(self, reg1, reg2):
50         m1 = map(None, reg1.getWoerter(),
51                   reg1.getTags(),
52                   reg1.getLemmata())
53         m2 = map(None, reg2.getWoerter(),
54                   reg2.getTags(),
55                   reg2.getLemmata())
56
57         if m1 == m2:
58             self.aiout.write("identisch\n<SATZ>\n")
59             return 0.0
60         self.aiout.write("verschieden\n<SATZ>\n")
61         return 1.0
62 #}
63
64
65 class AbstandChurchAndGale(Abstand): #{
66
67     def __repr__(self):
68         return "<AbstandChurchAndGale>"
69
70     def __init__(self, c = 1.0, s2 = 6.8):
71         Abstand.__init__(self)
72         self.c = c
73         self.s2 = s2
74
75     def zwischen(self, reg1, reg2):
76         len1 = len(reg1.getText())
77         len2 = len(reg2.getText())
78         return self.p_zwischen(len1, len2)
79
80     def p_zwischen(self, len1, len2):
81         if len1==0 and len2==0:
82             self.aiout.write("Laengen null\n<SATZ>\n")

```



```

83     return 0
84     c = self.c
85     s2 = self.s2
86     mean = (len1 + len2/c)/2
87     z = (c * len1 - len2)/math.sqrt(s2 * mean)    # sqrt(s2 * len1) ?
88     self.aiout.write("z = %s\n" % `z`)
89     if z < 0:
90         z = -z
91     pd = 2 * (1 - self.pnorm(z))                # uebernommen aus align.c
92     self.aiout.write("<SATZ>\n")
93     return 1.0 - pd
94     #if pd<0.00000001:
95     # pd= 0.00000001
96     #return -math.log(pd)
97
98     def pnorm(self, z):                          # uebernommen aus align.c
99         "Returns the area under a normal distribution"
100        "from -inf to z standard deviations"
101        t = 1.0/(1.0 + 0.2316419 * z)
102        pd = 1.0 - 0.3989423 * \
103            math.exp(-z * z/2) * \
104            (((1.330274429 * t - 1.821255978) * t \
105              + 1.781477937) * t - 0.356563782) * t + 0.319381530) * t
106        # see Abramowitz, M., and I. Stegun (1964), 26.2.17 p. 932
107        return pd
108    #}
109
110    class AbstandTest(Abstand): #{
111
112        def __init__(self):
113            Abstand.__init__(self)
114
115        def __repr__(self):
116            return "<AbstandTest>"
117
118        def zwischen(self, reg1, reg2):
119            len1 = len(reg1.getText())
120            len2 = len(reg2.getText())
121            d = len1 - len2
122            self.aiout.write("d = %s\n" % `d`)
123            if d < 0:
124                d = -d
125            if d > 10:
126                d = 10
127            self.aiout.write("<SATZ>\n")
128            return 0.1 * d
129
130    #}
131
132    class AbstandBidirectional(Abstand):
133
134        def __repr__(self):
135            return "<AbstandBidirectional>"
136
137        def __init__(self, d2e, e2d):
138            Abstand.__init__(self)
139            self.d2e = d2e
140            self.e2d = e2d
141
142        def close(self):
143            self.d2e.close()

```

```

144         self.e2d.close()
145
146     def zwischen(self, reg1, reg2):
147         a = self.d2e.zwischen(reg1, reg2)
148         b = self.e2d.zwischen(reg2, reg1)
149         self.aiout.write("Richtung d2e = %s\n" % 'a')
150         self.aiout.write("Richtung e2d = %s\n" % 'b')
151         self.aiout.write("<SATZ>\n")
152         return 0.5 * a + 0.5 * b
153
154
155 class AbstandWB(AbstandBidirectional):
156
157     def __repr__(self):
158         return "<AbstandWB>"
159
160     def __init__(self):
161         d2e = AbstandWBUndirectional('de', 'en')
162         e2d = AbstandWBUndirectional('en', 'de')
163         AbstandBidirectional.__init__(self, d2e, e2d)
164
165
166 class AbstandWBUndirectional(Abstand): #{
167
168     def __repr__(self):
169         return "<AbstandWBUndirectional>"
170
171     def __init__(self, quellsprache, zielsprache):
172         Abstand.__init__(self)
173         self.quellsprache = quellsprache
174         self.zielsprache = zielsprache
175         relevant1 = filterwb.RelevantNachTagP(zielsprache,0,1) # max = 1
176         self.wbf = filterwb.WoerterbuchFilter(quellsprache, zielsprache, relevant1)
177         self.relevant = filterwb.RelevantNachTagP(zielsprache)
178         self.atg = AbstandTG_2(3)
179
180     def close(self):
181         self.wbf.close()
182         self.atg.close()
183
184     def p_reduzierteWoerter(self, wliste):
185         liste = []
186         for wort in wliste:
187             nwort = string.lower(wort)
188             nwort = string.replace(nwort, ".", "")
189             nwort = string.replace(nwort, ",", "")
190             nwort = string.replace(nwort, "-", "")
191             liste.append(nwort)
192         return liste
193
194     def zwischen(self, reg1, reg2):
195         wb = self.wbf[reg1]
196         a1 = reg1.getRegionenMit(atom.Atom)
197         a2 = reg2.getRegionenMit(atom.Atom)
198         l1 = reg1.getLemmata()
199         l2 = reg2.getLemmata()
200         w1 = reg1.getWoerter()
201         w2 = reg2.getWoerter()
202         wr1 = self.p_reduzierteWoerter(w1)
203         wr2 = self.p_reduzierteWoerter(w2)
204         uebrig1 = len(a1) * [1]

```

```

205 uebrig2 = len(a2) * [1]
206 c1 = {}
207 c2 = {}
208 if l1 == []:
209     self.aiout.write("Region 1 leer.\n<SATZ>\n")
210     return 0.0
211 for index in range(0, len(a1)): #{
212     if wb.has_key(index): #{           # Relevant?
213         lemma = l1[index]
214         if lemma != "<unknown>": #{
215             c1[lemma] = l1.count(lemma)
216             ueabsatz = wb[index]
217             count2 = 0
218             if len(ueabsatz) > 0:
219                 uebrig1[index] = 0
220                 uemenge = {}
221                 ueatome = ueabsatz.getRegionenMit(atom.Atom)
222                 for ueatom in ueatome:
223                     if self.relevant.relevant(ueatom):
224                         ueatomLemma = ueatom.getLemmata()[0]
225                         if ueatomLemma != "<unknown>":
226                             uemenge[ueatomLemma] = 1
227                 self.aiout.write(''+lemma+' -> ''+string.join(uemenge.keys(),',', '')+'\n')
228                 for ue in uemenge.keys():
229                     try:
230                         pos = l2.index(ue)
231                         uebrig2[pos] = 0
232                         count2 = count2 + 1
233                     except ValueError:
234                         pass
235             #}
236         else: #{           # Sonderbehandlung: Wort anstatt Lemma messen
237             try:
238                 pos = wr2.index(wr1[index])
239                 uebrig2[pos] = 0
240                 count2 = count2 + 1
241             except ValueError:
242                 pass
243             #minimum = 1.0
244             #for i in range(0, len(a2)):
245             #    temp = self.atg(a1[index], a2[i])
246             #    if temp < minimum:
247             #        minimum = temp
248             #count2 = 1.0 - temp
249             #}
250             c2[lemma] = count2
251         #}
252     #}
253 #}
254 uebriglatome = []
255 for index in range(0, len(a1)):
256     if uebrig1[index]:
257         uebriglatome.append(a1[index])
258 uebriglatome = region.Regionen(uebriglatome)
259 uebrig2atome = []
260 for index in range(0, len(a2)):
261     if uebrig2[index]:
262         uebrig2atome.append(a2[index])
263 uebrig2atome = region.Regionen(uebrig2atome)
264 atgAbstand = self.atg(uebriglatome, uebrig2atome)
265 atgAnteil = len(uebriglatome)

```

```

266     insgesamt = len(a1)
267     awbAnteil = insgesamt - atgAnteil
268     x = 0.0
269     h2 = 0
270     for i in c1.keys():
271         t = float(c2[i]) / float(c1[i])
272         x = x + t
273         self.aiout.write('(i, c1[i], c2[i], t)'+'\n')
274         if c2[i]>0:
275             h2 = h2 + 1
276     n = len(c1.keys())
277     if n == 0:
278         self.aiout.write("Region 1 hat keine relevanten Woerter.\n<SATZ>\n")
279         return 0.0
280     x = x / n
281     self.aiout.write("Mittelwert " + 'x' + " - benutze aber 1.0"+'\n')
282     x = 1.0
283     v = 0.0
284     for i in c1.keys():
285         t = (float(c2[i]) / float(c1[i]))
286         t = t - x
287         v = v + t * t
288     v = v / n
289     #2do Kennzahl v umrechnen in Wahrscheinlichkeit fuer Entsprechung
290     # von reg1 und reg2
291     awbAbstand = 1.0 - math.exp(-v)
292     awbAbstand = 1.0 - float(h2) / float(len(c1.keys()))
293     self.aiout.write("atg " + 'atgAnteil' + 'atgAbstand'+'\n')
294     self.aiout.write("awb " + 'awbAnteil' + 'awbAbstand'+'\n')
295     self.aiout.write("<SATZ>\n")
296     return (atgAnteil * atgAbstand + awbAnteil * awbAbstand) / float(insgesamt)
297
298
299 class AbstandTG(Abstand): #{
300
301     def __init__(self, laenge = 3):
302         Abstand.__init__(self)
303         self.laenge = laenge
304
305     def __repr__(self):
306         return "<AbstandTG>"
307
308     def zwischen(self, reg1, reg2):
309         t1 = " " + reg1.getText() + " "
310         t2 = " " + reg2.getText() + " "
311         # self.aiout.write(("---"+reg1.getText()+"/"+reg2.getText()+ "---")+'\n')
312         # self.aiout.write(("---"+t1+"/"+t2+"---")+'\n')
313         if t1 == "":
314             self.aiout.write("leere Quellregion"+'\n')
315             self.aiout.write("<SATZ>\n")
316             return 0.0
317         # self.aiout.write(("---"+t1+"/"+t2+"---")+'\n')
318         c1 = self.counttg(t1)
319         c2 = self.counttg(t2)
320         self.aiout.write("c1:" + 'c1' + '\n')
321         self.aiout.write("c2:" + 'c2' + '\n')
322         relevant = c1.keys()
323         for i in c1.keys():
324             if not c2.has_key(i):
325                 c2[i] = 0
326         x = 0.0

```

```

327 h2 = 0
328 for i in relevant:
329     t = float(c2[i]) / float(c1[i])
330     x = x + t
331     self.aiout.write('\i' + '\t' + 'c1[i]' + '\t' + 'c2[i]' + '\t' + '\n')
332     if c2[i] > 0:
333         h2=h2+1
334 n = len(relevant)
335 if n == 0:
336     self.aiout.write("keine %s-gramme in Region 1\n<SATZ>\n" %self.laenge)
337     return 0.0
338 #x = x / n
339 x = 1.0
340 v = 0.0
341 for i in relevant:
342     t = (float(c2[i]) / float(c1[i]))
343     t = t - x
344     v = v + t * t
345 v = v / n
346 #2do Kennzahl v umrechnen in Wahrscheinlichkeit fuer Entsprechung
347 # von reg1 und reg2
348 atgAbstand = 1.0 - math.exp(-v)
349 # atgAbstand = 1.0 - float(h2) / float(len(relevant))
350 self.aiout.write("<SATZ>\n")
351 return atgAbstand
352
353 def counttg(self, text):
354     c = {}
355     for i in range(0, len(text)-self.laenge+1):
356         tg = text[i:i+self.laenge]
357         if not c.has_key(tg):
358             c[tg] = 1
359         else:
360             c[tg] = c[tg] + 1
361     return c
362
363 #}
364
365 class AbstandTG_2(Abstand): #{
366
367     def __init__(self, laenge = 3):
368         Abstand.__init__(self)
369         self.laenge = laenge
370         self.cache = cache.DefaultCache(self)
371
372     def __repr__(self):
373         return "<AbstandTG_2>"
374
375     def p_reduziert(self, wort):
376         nwort = string.lower(wort)
377         for i in [".", ",", ":", "$"]:
378             nwort = string.replace(nwort, i, "")
379         for i in ["-", "/"]:
380             nwort = string.replace(nwort, i, " ")
381         return " " + nwort + " "
382
383     def zwischen(self, reg1, reg2):
384         t1 = self.p_reduziert(reg1.getText())
385         t2 = self.p_reduziert(reg2.getText())
386         c1 = self.counttg(t1)
387         c2 = self.counttg(t2)

```

```

388     key1 = c1.keys()
389     key2 = c2.keys()
390     last = ""
391     keys = key1 + key2
392     keys.sort()
393     gemeinsam = 0
394     insgesamt = 0
395     for key in keys:
396         if last == key:
397             continue
398         if c1.has_key(key):
399             num1 = c1[key]
400         else:
401             num1 = 0
402         if c2.has_key(key):
403             num2 = c2[key]
404         else:
405             num2 = 0
406         gemeinsam = gemeinsam + min([num1,num2])
407         insgesamt = insgesamt + max([num1,num2])
408         last = key # siehe Doku im Abschlussbericht
409     if insgesamt > 0:
410         v = float(gemeinsam) / float(insgesamt)
411     else:
412         v = 1.0
413     atgAbstand = 1.0 - v
414     self.aiout.write("gemeinsam,insgesamt = %s\n" %'(gemeinsam, insgesamt)')
415     self.aiout.write("'%s' - '%s'\n" %(t1,t2))
416     self.aiout.write("<SATZ>\n")
417     return atgAbstand
418
419 def counttg(self, text):
420     ckey = text
421     if self.cache.has_key(ckey):
422         retval = self.cache[ckey]
423         # sys.stderr.write("TG_2: %s aus cache\n" %'ckey')
424     else:
425         c = {}
426         for i in range(0, len(text)-self.laenge+1):
427             tg = text[i:i+self.laenge]
428             if not c.has_key(tg):
429                 c[tg] = 1
430             else:
431                 c[tg] = c[tg] + 1
432         retval = c
433         self.cache[ckey] = retval
434     return retval
435
436
437 class NullFile:
438     def write(self, data):
439         pass
440
441 class AbstandWB_2(Abstand, zeitmessung.Zeitmessung):
442
443     def __repr__(self):
444         if self.nurNomen:
445             s = "<AbstandWB_2 %s nach %s (nur Nomen)>"
446         else:
447             s = "<AbstandWB_2 %s nach %s>"
448         return s %(self.quellsprache,self.zielsprache)

```

```

449
450 def __init__(self, quellsprache = "de", zielsprache = "en", nurNomen = 0):
451     zeitmessung.Zeitmessung.__init__(self)
452     Abstand.__init__(self)
453     self.quellsprache = quellsprache
454     self.zielsprache = zielsprache
455     if nurNomen:
456         temp = filterwb.RelevantWegenNomen(quellsprache,0,1) # max = 1
457     else:
458         temp = filterwb.RelevantNachTagP(quellsprache, 0, 1)
459     self.wbf1 = filterwb.WoerterbuchFilter(quellsprache, zielsprache, temp)
460     if nurNomen:
461         temp = filterwb.RelevantWegenNomen(zielsprache,0,1) # max = 1
462     else:
463         temp = filterwb.RelevantNachTagP(zielsprache, 0, 1)
464     self.wbf2 = filterwb.WoerterbuchFilter(zielsprache, quellsprache, temp)
465     self.nurNomen = nurNomen
466     self.atg = AbstandTG_2(3)
467     self.acg = AbstandChurchAndGale()
468     self.cache = cache.DefaultCache(self)
469
470 def close(self):
471     self.wbf1.close()
472     self.wbf2.close()
473     self.atg.close()
474     self.acg.close()
475
476 def statistic(self):
477     return { "wbf1" : self.wbf1.statistic(),
478             "wbf2" : self.wbf2.statistic(),
479             "wb1"  : self.wbf1.wb.statistic(),
480             "wb2"  : self.wbf2.wb.statistic(),
481             "self" : zeitmessung.Zeitmessung.statistic(self),
482             }
483
484 def p_daten(self, reg1, wbf1):
485     self.p_begin("p_daten")
486     ckey = ('wbf1',reg1.getText()) # 2do
487     if self.cache.has_key(ckey):
488         retval = self.cache[ckey]
489         # sys.stderr.write("WB_2: %s aus cache\n" % `ckey`)
490     else:
491         wb1 = wbf1[reg1]
492         a1 = reg1.getRegionenMit(atom.Atom)
493         l1 = reg1.getLemmata()
494         w1 = reg1.getWoerter()
495         wr1 = None # self.p_reduzierteWoerter(w1)
496         i1 = len(a1) * [0] # 0 = 'zu bearbeiten'
497         relevant1 = []
498         for index in range(len(i1)):
499             if wb1.has_key(index): # relevant?
500                 #lemmal = l1[index]
501                 relevant1.append(index)
502             else:
503                 i1[index] = 1 # als irrelevant markieren
504         retval = (wb1, a1, l1, w1, wr1, i1, relevant1)
505         self.cache[ckey] = retval
506     (wb1, a1, l1, w1, wr1, i1, relevant1) = retval
507     i1 = i1[:] # Kopie erstellen
508     self.p_end("p_daten")
509     return (wb1, a1, l1, w1, wr1, i1, relevant1)

```

```

510
511 def zwischen(self, reg1, reg2):
512     self.p_begin("zwischen")
513     (wb1, a1, l1, w1, wr1, i1, relevant1) = self.p_daten(reg1, self.wbf1)
514     (wb2, a2, l2, w2, wr2, i2, relevant2) = self.p_daten(reg2, self.wbf2)
515     aiextral = StringIO.StringIO()
516     aiextra2 = StringIO.StringIO()
517     temp = self.aiout
518     self.aiout = aiextral
519     self.p_setUe(relevant1, relevant2, wb1, l1, l2, i1, i2)
520     self.aiout = aiextra2
521     self.p_setUe(relevant2, relevant1, wb2, l2, l1, i2, i1)
522     self.aiout = temp
523     self.aiout.write("infol = %s\n" %`map(None, i1, w1)`')
524     self.aiout.write("info2 = %s\n" %`map(None, i2, w2)`')
525     uenum1 = self.p_uenum(relevant1, i1)
526     uenum2 = self.p_uenum(relevant2, i2)
527     gemeinsam = min([uenum1, uenum2])
528     insgesamt = max([uenum1, uenum2])
529     self.aiout.write("AWB_2: g,i = %s\n" %`(gemeinsam,insgesamt)`')
530     if insgesamt > 0:
531         awbAbstand = 1.0 - float(gemeinsam) / float(insgesamt)
532     else:
533         awbAbstand = 1.0 # 2do: Bewertung von 'insgesamt' verbessern
534     awbAnteil = uenum1 + uenum2
535     uebrig1 = self.p_uebrig(a1, relevant1, i1)
536     uebrig2 = self.p_uebrig(a2, relevant2, i2)
537     self.aiout.write("uebrig1 = %s\n" %uebrig1.getText())
538     self.aiout.write("uebrig2 = %s\n" %uebrig2.getText())
539     atgAbstand = self.atg(uebrig1, uebrig2)
540     atgAnteil = len(uebrig1) + len(uebrig2)
541     regcg1 = len(i1) - uenum1 - len(uebrig1)
542     if regcg1 < 0:
543         sys.stderr.write("Fehler: |i1|-uenum1-|uebrig1| = %s\n" %regcg1)
544     regcg2 = len(i2) - uenum2 - len(uebrig2)
545     if regcg2 < 0:
546         sys.stderr.write("Fehler: |i2|-uenum2-|uebrig2| = %s\n" %regcg2)
547     acgAnteil = regcg1 + regcg2
548     acgAbstand = self.acg.p_zwischen(regcg1, regcg2) # Hack
549     # 2do: eigentlich muesste man nicht die Anzahl der Woerter
550     # sondern die Anzahl der Zeichen an acg uebergeben
551     self.aiout.write("len(regcg1) = %s\n" %regcg1)
552     self.aiout.write("len(regcg2) = %s\n" %regcg2)
553     self.aiout.write("WB: wert, anteil = %s\n" %`(awbAbstand,awbAnteil)`')
554     self.aiout.write("TG: wert, anteil = %s\n" %`(atgAbstand,atgAnteil)`')
555     self.aiout.write("CG: wert, anteil = %s\n" %`(acgAbstand,acgAnteil)`')
556     bonus = max([len(i1),len(i2)])
557     if awbAnteil > 0:
558         awbAnteil = awbAnteil + bonus
559         if atgAnteil > 0:
560             awbAnteil = awbAnteil + bonus
561         self.aiout.write("WB-Anteil erhoehrt auf %s\n" %awbAnteil)
562     if atgAnteil > 0:
563         atgAnteil = atgAnteil + bonus
564         self.aiout.write("TG-Anteil erhoehrt auf %s\n" %atgAnteil)
565     gesamt = awbAnteil + atgAnteil + acgAnteil
566     if gesamt > 0:
567         retval = float( awbAnteil * awbAbstand +\
568                        atgAnteil * atgAbstand +\
569                        acgAnteil * acgAbstand ) / float( gesamt )
570     else:

```



```

571         retval = 0.0
572     self.aiout.write("Abstand insgesamt = %s\n" %retval)
573     aiextral.seek(0)
574     self.aiout.write("--- Ue-Info (1):\n")
575     self.aiout.write(aiextral.read())
576     aiextra2.seek(0)
577     self.aiout.write("--- Ue-Info (2):\n")
578     self.aiout.write(aiextra2.read())
579     self.aiout.write("\n<SATZ>\n")
580     self.p_end("zwischen")
581     return retval
582
583 def p_uebrig(self, atome, relevant, info):
584     self.p_begin("p_uebrig")
585     uebrig = []
586     for index in relevant:
587         if (info[index] & 2) == 0:      # noch nicht bearbeitet?
588             uebrig.append(atome[index])
589     retval = region.Regionen(uebrig)
590     self.p_end("p_uebrig")
591     return retval
592
593 def p_setUe(self, relevant1, relevant2, wb1, l1, l2, i1, i2):
594     self.p_begin("p_setUe")
595     for index1 in relevant1:
596         if wb1.has_key(index1):          # relevant?
597             lemmal = l1[index1]
598             uel = wb1[index1]
599             uellemmata = uel.getLemmata() # leer wenn lemmal '<unknown>'
600             #if not uellemmata:          # leer? # 2do
601             #     continue                # dann aussichtslos
602         for index2 in relevant2:
603             lemma2 = l2[index2]
604             if lemma2 == "<unknown>":
605                 pass
606             # 2do: Woerter vergleichen
607             # wort2[index2] in uel.getWoerter() ?
608             elif lemma2 in uellemmata:
609                 i1[index1] = i1[index1] | 2 # markieren
610                 i2[index2] = i2[index2] | 2
611                 self.aiout.write("%s -> %s\n" %(lemmal,uellemmata))
612                 self.aiout.write("index %s -> %s\n" %(index1,index2))
613             #else: # 2do: klaeren, ob relevant1 nich in p_daten entspr. init.
614             #     sys.stderr.write("if doch notwendig\n")
615     self.p_end("p_setUe")
616
617 def p_uenum(self, relevant, info):      # zaehlt die markierten Woerter
618     self.p_begin("p_uenum")
619     num = 0
620     for index in relevant:
621         if (info[index] & 2) != 0:
622             num = num + 1
623     self.p_end("p_uenum")
624     return num
625
626 import wb
627 import filterwb
628 import eingabe
629 from DatabaseAPI import config, db, regiondb, cache
630 from Region import atom, absatz, region
631

```

```

632 def main():
633     dbconn = db.Db()
634     regiondb.setDefaultDbconn(dbconn)
635     ab = { "C&G" : AbstandChurchAndGale(),
636           "CG"  : AbstandChurchAndGale(),
637           "TEST" : AbstandTest(),
638           "NULL" : Abstand(0.0),
639           "EINS" : Abstand(1.0),
640           "WB"   : AbstandWB(),
641           "TG"   : AbstandTG(),
642           "WB2"  : AbstandWB_2(),
643           "TG2"  : AbstandTG_2(),
644           "WB2N" : AbstandWB_2("de","en",1), # nur Nomen
645     }
646     wahl = "WB2"
647     fout = sys.stdout
648     verbose = 0
649     text = 0
650     hilfe = len(sys.argv) == 1
651     reg = "segment"
652     dok = eingabe.Liste(verbose)
653     while len(sys.argv) > 1: #{
654         if verbose:
655             sys.stderr.write("args left: %s\n" %`sys.argv[1:]`)
656         if sys.argv[1] == "-h":
657             hilfe = 1
658             break
659         elif sys.argv[1] == "-u":
660             text = 1
661             del sys.argv[1]
662         elif sys.argv[1] == "-v":
663             verbose = 1
664             dok.setVerbose(verbose)
665             del sys.argv[1]
666         elif sys.argv[1][:2] == "-r":
667             reg = sys.argv[1][2:]
668             del sys.argv[1]
669             if not reg:
670                 try:
671                     reg = sys.argv[1]
672                     del sys.argv[1]
673                 except:
674                     hilfe = 1
675                     break
676             if verbose:
677                 sys.stderr.write("region type = %s\n" %wahl)
678         elif sys.argv[1][:2] == "-m":
679             wahl = sys.argv[1][2:]
680             del sys.argv[1]
681             if not wahl:
682                 try:
683                     wahl = sys.argv[1]
684                     del sys.argv[1]
685                 except:
686                     hilfe = 1
687                     break
688             if verbose:
689                 sys.stderr.write("distance measure = %s\n" %wahl)
690         elif sys.argv[1][:2] == "-o":
691             fname = sys.argv[1][2:]
692             del sys.argv[1]

```

```

693     if not fname:
694         try:
695             fname = sys.argv[1]
696             del sys.argv[1]
697         except:
698             hilfe = 1
699             break
700     fout = open(fname, "w")
701     if verbose:
702         sys.stderr.write("output to file %s\n" %fname)
703 else:
704     eaten = dok.eatOptions(sys.argv)
705     if not eaten:
706         hilfe = 1
707         break
708     if verbose:
709         sys.stderr.write("eingabe.Liste has eaten %s args\n" %eaten)
710 if hilfe:
711     if verbose:
712         sys.stderr.write("args left: %s\n" %`sys.argv[1:]`)
713     sys.stderr.write("Usage: %s <options>\n" %sys.argv[0])
714     sys.stderr.write("options:  -h          print out this help\n")
715     sys.stderr.write("          -m distmeas  choose distance measure:\n")
716     keys = ab.keys()
717     keys.sort()
718     for w in keys:
719         sys.stderr.write("          %s\t%s\n" %(w,`ab[w]`))
720     sys.stderr.write("          -o filename  write output to file\n")
721     sys.stderr.write("          -r regtype   one of\n")
722     for i in ["satz", "segment", "absatz", "dokument"]:
723         sys.stderr.write(25*' ' + "%s\n" %i)
724     sys.stderr.write("          -u          output index and text\n")
725     sys.stderr.write("          -v          verbose mode\n")
726     dok.help(sys.stderr)
727     dok.close()
728     dbconn.close()
729     for key in ab.keys():
730         ab[key].close()
731     sys.exit(1)
732 if verbose:
733     sys.stderr.write("distance measure %s = %s\n" %(wahl,`ab[wahl]`))
734 dok.finishOptions()
735 if dok.getQuellsprache() == "de" and dok.getZielsprache() == "en":
736     l1 = eingabe.getRegionen(dok, reg)
737     l2 = eingabe.getRegionen(dok.alignedRegion(), reg)
738 elif dok.getQuellsprache() == "en" and dok.getZielsprache() == "de":
739     l1 = eingabe.getRegionen(dok.alignedRegion(), reg)
740     l2 = eingabe.getRegionen(dok, reg)
741 elif not wahl in ["WB"]:          # Liste der sprachabhaengigen Abstandsmasse
742     l1 = eingabe.getRegionen(dok, reg)          # reihenfolge egal
743     l2 = eingabe.getRegionen(dok.alignedRegion(), reg)
744 else:
745     sys.stderr.write("Error: cannot compare %s " %dok.getQuellsprache())
746     sys.stderr.write("and %s " %dok.getZielsprache())
747     sys.stderr.write("with %s\n" %`ab[wahl]`)
748     sys.exit(1)
749 if verbose:
750     sys.stderr.write("number of regions: %s\n" %`(len(l1),len(l2))`)
751 for index in range(max(len(l1), len(l2))):
752     if text:
753         fout.write("--- %s Nr. %s ---\n" %(reg, index))

```

```

754         if l1.has_key(index):
755             fout.write(l1[index].getText()+'\n')
756         if l2.has_key(index):
757             fout.write(l2[index].getText()+'\n')
758         if l1.has_key(index) and l2.has_key(index):
759             wert = ab[wahl].zwischen(l1[index], l2[index])
760             fout.write("%s\n" % `wert`)
761         else:
762             fout.write("***\n")
763         if text:
764             fout.write('\n')
765     dok.close()
766     for key in ab.keys():
767         ab[key].close()
768     dbconn.close()
769
770 if __name__ == '__main__':
771     main()

```

M.3.2 align.sh

```

1  #!/bin/bash
2
3  # align.sh
4
5  # Aufruf: align.sh dokumentprefix abstandsmass
6  #  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
7
8  ALIGN=`dirname $0`
9
10 if [ "$7" ] ; then
11
12     #MASS="TG2"
13     MASS="WB2"
14     #LEAVE_OUT_DIST=2.5
15     #LEAVE_OUT_DIST_MAX=2.5
16     #LEAVE_OUT_DIST_GRADIENT=0.00
17     LEAVE_OUT_DIST=1.5
18     LEAVE_OUT_DIST_MAX=3.53
19     LEAVE_OUT_DIST_GRADIENT=0.12
20     test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
21     test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
22     test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
23     test "$3" = "<dokende>"      && HDEL="dokument"
24     test "$3" = "<DOKENDE>"      && HDEL="dokument"
25     test "$3" = "<ABSATZ>"       && HDEL="absatz"
26     test "$5" = "<ABSATZ>"       && SDEL="absatz"
27     test "$5" = "<segmentgrenze>" && SDEL="satz"
28     test "$5" = "<SATZ>"         && SDEL="satz"
29     test "$HDEL" || { echo "Harddelimiter $3 unbekannt" >&2 ; exit 1 ; }
30     test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
31
32     RED1=`dirname "$6" `/\`basename "$6" .de.tag`
33     RED2=`dirname "$7" `/\`basename "$7" .en.tag`
34     STAMM="$6"
35     test "$RED1" = "$RED2" && STAMM="$RED1"
36
37     # Version mit Abstandsmatrix
38
39     python "$ALIGN/nmatrix.py" -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_D
40

```

```

41 elif [ "$2" ] ; then
42
43     python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/mergemtr
44
45 else
46
47     echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
48     echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
49     exit 1
50
51 fi
52

```

M.3.3 align2al.py

```

1  #!/usr/bin/python
2
3  import os
4  import string
5  import sys
6  import types
7
8  from DatabaseAPI import dokumentdb, db, regiondb, config
9  from Region      import satz, dokument, segment, absatz
10
11 dbconn = db.Db()
12 regiondb.setDefaultDbconn(dbconn)
13
14 def main():
15     global dbconn
16     verbose = 0
17     ausgabeInsArbeitsverzeichnis = 0
18     hilfe = 0
19     while len(sys.argv) > 1: #{
20         if verbose:
21             sys.stderr.write("args left: %s\n" % `sys.argv[1:]`)
22         if sys.argv[1] == "-h":
23             hilfe = 1
24             break
25         elif sys.argv[1] == "-v":
26             verbose = 1
27             del sys.argv[1]
28         elif sys.argv[1] == "-t":
29             ausgabeInsArbeitsverzeichnis = 1
30             del sys.argv[1]
31         else:
32             break
33     if not hilfe and len(sys.argv) == 2:
34         fname = sys.argv[-1]
35         if fname == "-":
36             align = sys.stdin
37         else:
38             align = open(sys.argv[-1], "r")
39     else:
40         hilfe = 1
41     if hilfe:
42         sys.stderr.write("Benutzung: %s [-t] <.align-Datei>\n" % sys.argv[0])
43         sys.stderr.write("\t-t = Ausgabe ins Arbeitsverzeichnis\n")
44     else:
45         zeile = align.readline()
46         if zeile[:14] == "Dokument-DB-ID":

```

```

47         id = int(zeile[14:])
48         dok1 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["de"])
49         dok2 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["en"])
50         out1 = open("db_"+'id'+".de.tag.al", "w")
51         out2 = open("db_"+'id'+".en.tag.al", "w")
52     elif zeile[:14] == "Dokument-Datei":
53         name = string.strip(zeile[15:])
54         dok1 = dokument.DokumentVonDatei(name+".de.tag")
55         dok2 = dokument.DokumentVonDatei(name+".en.tag")
56         if ausgabeInsArbeitsverzeichnis:
57             name = os.path.basename(name)
58             out1 = open(name+".de.tag.al", "w")
59             out2 = open(name+".en.tag.al", "w")
60     elif zeile[:19] == "Paar Dokument-Datei":
61         pos = string.find(zeile, " und Dokument-Datei")
62         name1 = string.strip(zeile[20:pos])
63         name2 = string.strip(zeile[pos+20:])
64         dok1 = dokument.DokumentVonDatei(name1)
65         dok2 = dokument.DokumentVonDatei(name2)
66         if ausgabeInsArbeitsverzeichnis:
67             name1 = os.path.basename(name1)
68             name2 = os.path.basename(name2)
69             out1 = open(name1+".al", "w")
70             out2 = open(name2+".al", "w")
71     elif zeile[:6] == "Absatz":
72         raise NotImplementedError
73     else:
74         raise Exception, "Dokumenttyp nicht identifizierbar: %s" %'zeile'
75     zeile = align.readline()
76     if string.lower(zeile) == "nach satz\n":
77         l1 = dok1.getRegionenMit(satz.Satz)
78         l2 = dok2.getRegionenMit(satz.Satz)
79         segclass = segment.Segment
80     elif string.lower(zeile[:5]) == "nach ":
81         raise NotImplementedError
82     else:
83         raise Exception
84     zeile = align.readline()
85     while zeile != "":
86         zeilenliste = string.split(zeile, "\t") # Zeile am Tab trennen
87         if verbose:
88             sys.stderr.write("Verarbeite Segment Nr. %s\n" %zeilenliste[0])
89         slicel = eval(zeilenliste[1])
90         slice2 = eval(zeilenliste[2])
91         if verbose:
92             sys.stderr.write("%s zu %s\n" %('\slicel', '\slice2'))
93         seg1 = segclass(l1[slicel[0]:slicel[-1]+1])
94         seg2 = segclass(l2[slice2[0]:slice2[-1]+1])
95         seg1.writeTagzeilen(out1.write)
96         seg2.writeTagzeilen(out2.write)
97         zeile = align.readline()
98     align.close()
99     out1.close()
100    out2.close()
101    dbconn.close()
102
103 def dokPaarUndAusgabe(zeile, ausgabeInsArbeitsverzeichnis):
104     if zeile[:14] == "Dokument-DB-ID":
105         id = int(zeile[14:])
106         dok1 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["de"])
107         dok2 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["en"])

```

```

108     out1 = open("db_"+`id`+".de.tag.al", "a")
109     out2 = open("db_"+`id`+".en.tag.al", "a")
110     elif zeile[:14] == "Dokument-Datei":
111         name = string.strip(zeile[15:])
112         dok1 = dokument.DokumentVonDatei(name+".de.tag")
113         dok2 = dokument.DokumentVonDatei(name+".en.tag")
114         if ausgabeInsArbeitsverzeichnis:
115             name = os.path.basename(name)
116             out1 = open(name+".de.tag.al", "a")
117             out2 = open(name+".en.tag.al", "a")
118     elif zeile[:6] == "Absatz":
119         pos = string.find(zeile, "von")
120         index = int(zeile[6:pos])
121         temp = dokPaarUndAusgabe(zeile[pos+4:], ausgabeInsArbeitsverzeichnis)
122         if not temp:
123             sys.stderr.write("Dokument zu Absatz nicht angegeben\n")
124             return None
125         dok1 = temp[0].getRegionenMit(absatz.Absatz)[index]
126         dok2 = temp[1].getRegionenMit(absatz.Absatz)[index]
127         out1 = temp[2]
128         out2 = temp[3]
129     else:
130         return None
131     return (dok1, dok2, out1, out2)
132
133 if __name__ == '__main__':
134     try:
135         main()
136     except:
137         import traceback
138         import StringIO
139         tb = StringIO.StringIO()
140         traceback.print_exc(file=tb)
141         tb.seek(0)
142         sys.stderr.write("in %s:\n" %sys.argv[0])
143         zeile = tb.readline()
144         while zeile != "":
145             sys.stderr.write("\t"+zeile)
146             zeile = tb.readline()
147         dbconn.close()
148         sys.exit(1)

```

M.3.4 align_lzu1.sh

```

1  #!/bin/bash
2
3  # align_lzu1.sh
4
5  # Aufruf: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
6
7  ALIGN=`dirname $0`
8
9  if [ "$7" ] ; then
10
11     test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
12     test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
13     test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
14     test "$3" = "<dokende>"      && HDEL="dokument"
15     test "$3" = "<DOKENDE>"      && HDEL="dokument"
16     test "$3" = "<ABSATZ>"       && HDEL="absatz"
17     test "$5" = "<ABSATZ>"       && SDEL="absatz"

```

```

18 test "$5" = "<segmentgrenze>" && SDEL="satz"
19 test "$5" = "<SATZ>" && SDEL="satz"
20 test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
21 test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
22
23 RED1=`fgrep "<segmentgrenze>" "$6" | wc -l`
24 RED2=`fgrep "<segmentgrenze>" "$7" | wc -l`
25 test "$RED1" = "$RED2" && echo "Warnung: Anzahl Segmente verschieden: $RED1 $RED2" >&2
26
27 # Version, die 1:1 Alignment vornimmt
28 # Achtung: Es wird nicht ueberprueft, ob die Anzahl der
29 # Segmente ueberein stimmt.
30
31 cp "$6" "$6.al"
32 cp "$7" "$7.al"
33
34 else
35
36 echo "Aufruf: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
37 exit 1
38
39 fi
40

```

M.3.5 align_ai.sh

```

1 #!/bin/bash
2
3 # align.sh
4
5 # Aufruf: align.sh dokumentprefix abstandsmass
6 # oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
7
8 ALIGN=`dirname $0`
9
10 if [ "$7" ] ; then
11
12 #MASS="TG2"
13 MASS="WB2"
14 #LEAVE_OUT_DIST=2.5
15 #LEAVE_OUT_DIST_MAX=2.5
16 #LEAVE_OUT_DIST_GRADIENT=0.00
17 LEAVE_OUT_DIST=1.5
18 LEAVE_OUT_DIST_MAX=3.53
19 LEAVE_OUT_DIST_GRADIENT=0.12
20 test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
21 test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
22 test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
23 test "$3" = "<dokende>" && HDEL="dokument"
24 test "$3" = "<DOKENDE>" && HDEL="dokument"
25 test "$3" = "<ABSATZ>" && HDEL="absatz"
26 test "$5" = "<ABSATZ>" && SDEL="absatz"
27 test "$5" = "<segmentgrenze>" && SDEL="satz"
28 test "$5" = "<SATZ>" && SDEL="satz"
29 test "$HDEL" || { echo "Hardelimiter $3 unbekannt" >&2 ; exit 1 ; }
30 test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
31
32 RED1=`dirname "$6"``basename "$6" .de.tag`
33 RED2=`dirname "$7"``basename "$7" .en.tag`
34 STAMM="$6"
35 test "$RED1" = "$RED2" && STAMM="$RED1"

```



```

36
37 # Version mit Abstandmatrix und -info Ausgabe
38
39 python "$ALIGN/nmatrix.py" -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_DIST
40 python "$ALIGN/mergemtr.py" "$STAMM".ai_pre "$STAMM".abstandinfo < "$STAMM".mtr_pre | tee "$STAMM".mtr |
41 java -jar "$ALIGN/al.jar" |
42 python "$ALIGN/align2al.py" - && rm "$STAMM".ai_pre "$STAMM".mtr_pre
43
44 elif [ "$2" ] ; then
45
46     python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/mergemtr
47
48 else
49
50     echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
51     echo " oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
52     exit 1
53
54 fi
55

```

M.3.6 align_debug.sh

```

1 #!/bin/bash
2
3 # align.sh
4
5 # Aufruf: align.sh dokumentprefix abstandsmass
6 # oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
7
8 ALIGN=`dirname $0`
9
10 if [ "$7" ] ; then
11
12     #MASS="TG2"
13     MASS="WB2"
14     #LEAVE_OUT_DIST=2.5
15     #LEAVE_OUT_DIST_MAX=2.5
16     #LEAVE_OUT_DIST_GRADIENT=0.00
17     LEAVE_OUT_DIST=1.5
18     LEAVE_OUT_DIST_MAX=3.53
19     LEAVE_OUT_DIST_GRADIENT=0.12
20     test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
21     test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
22     test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
23     test "$3" = "<dokende>"      && HDEL="dokument"
24     test "$3" = "<DOKENDE>"      && HDEL="dokument"
25     test "$3" = "<ABSATZ>"       && HDEL="absatz"
26     test "$5" = "<ABSATZ>"       && SDEL="absatz"
27     test "$5" = "<segmentgrenze>" && SDEL="satz"
28     test "$5" = "<SATZ>"         && SDEL="satz"
29     test "$HDEL" || { echo "Harddelimiter $3 unbekannt" >&2 ; exit 1 ; }
30     test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
31
32     RED1=`dirname "$6" `/\`basename "$6" .de.tag`
33     RED2=`dirname "$7" `/\`basename "$7" .en.tag`
34     STAMM="$6"
35     test "$RED1" = "$RED2" && STAMM="$RED1"
36
37 # Version mit vollstaendiger Ausgabe
38

```

```

39     python "$ALIGN/nmatrix.py" -v -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_DIST_GRADIENT"
40     python "$ALIGN/mergemtr.py" "$STAMM".ai_pre "$STAMM".abstandinfo < "$STAMM".mtr_pre | tee "$STAMM".mtr |
41     java -jar "$ALIGN/al.jar" | tee "$STAMM".align |
42     python "$ALIGN/align2al.py" -
43
44 elif [ "$2" ] ; then
45
46     python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/merge.py" "$2"
47
48 else
49
50     echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
51     echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
52     exit 1
53
54 fi
55

```

M.3.7 align_mtr.sh

```

1  #!/bin/bash
2
3  # align_mtr.sh
4
5  # Aufruf: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
6
7  ALIGN='dirname $0'
8
9  if [ "$7" ] ; then
10
11     test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
12     test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
13     test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
14     test "$3" = "<dokende>"      && HDEL="dokument"
15     test "$3" = "<DOKENDE>"      && HDEL="dokument"
16     test "$3" = "<ABSATZ>"       && HDEL="absatz"
17     test "$5" = "<ABSATZ>"       && SDEL="absatz"
18     test "$5" = "<segmentgrenze>" && SDEL="satz"
19     test "$5" = "<SATZ>"         && SDEL="satz"
20     test "$HDEL" || { echo "Harddelimiter $3 unbekannt" >&2 ; exit 1 ; }
21     test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
22
23     RED1='dirname "$6"'\`basename "$6" .de.tag`
24     RED2='dirname "$7"'\`basename "$7" .en.tag`
25     STAMM="$6"
26     test "$RED1" = "$RED2" && STAMM="$RED1"
27
28     # Version, die Abstandsmatrix .mtr voraussetzt
29
30     test -f "$STAMM".mtr || { echo "Abstandsmatrix $STAMM.mtr nicht gefunden" >&2 ; exit 1 ; }
31
32     java -jar -Xmx=650m "$ALIGN/al.jar" < "$STAMM".mtr | python "$ALIGN/align2al.py" -
33
34 else
35
36     echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
37     echo "  oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
38     exit 1
39
40 fi
41

```

M.3.8 align_pur.sh

```

1 #!/bin/bash
2
3 # align.sh
4
5 # Aufruf: align.sh dokumentprefix abstandsmass
6 # oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en
7
8 ALIGN=`dirname $0`
9
10 if [ "$7" ] ; then
11
12     #MASS="TG2"
13     MASS="WB2"
14     #LEAVE_OUT_DIST=2.5
15     #LEAVE_OUT_DIST_MAX=2.5
16     #LEAVE_OUT_DIST_GRADIENT=0.00
17     LEAVE_OUT_DIST=1.5
18     LEAVE_OUT_DIST_MAX=3.53
19     LEAVE_OUT_DIST_GRADIENT=0.12
20     test "$1" = "-e" || { echo "arg1 muss -e sein" >&2 ; exit 1 ; }
21     test "$2" = "-D" || { echo "arg2 muss -D sein" >&2 ; exit 1 ; }
22     test "$4" = "-d" || { echo "arg4 muss -d sein" >&2 ; exit 1 ; }
23     test "$3" = "<dokende>"      && HDEL="dokument"
24     test "$3" = "<DOKENDE>"      && HDEL="dokument"
25     test "$3" = "<ABSATZ>"        && HDEL="absatz"
26     test "$5" = "<ABSATZ>"        && SDEL="absatz"
27     test "$5" = "<segmentgrenze>" && SDEL="satz"
28     test "$5" = "<SATZ>"          && SDEL="satz"
29     test "$HDEL" || { echo "Harddelimiter $3 unbekannt" >&2 ; exit 1 ; }
30     test "$SDEL" || { echo "Softelimiter $5 unbekannt" >&2 ; exit 1 ; }
31
32     # Version nur fuer .al
33
34     python "$ALIGN/nmatrix.py" -m "$MASS" -l "$LEAVE_OUT_DIST" -L "$LEAVE_OUT_DIST_GRADIENT" -M "$LEAVE_OUT_DIST"
35
36 elif [ "$2" ] ; then
37
38     python "$ALIGN/nmatrix.py" -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" | python "$ALIGN/mergentr
39
40 else
41
42     echo "Aufruf: align.sh dokumentprefix abstandsmass" >&2
43     echo " oder: align.sh -e -D harddelimiter -d softdelimiter dok-de dok-en" >&2
44     exit 1
45
46 fi
47

```

M.3.9 alignment.py

```

1 #!/usr/bin/python
2
3 # Kommandozeilentool alignment: List Jobanzahl + Abstandsmatrizen von stdin und
4 # gibt Ergebnis von Patricks MatrixAligner aus.
5
6 import popen2
7 import string
8 import sys
9

```

```

10 from DatabaseAPI import config
11
12 def main():
13     anzahl = int(string.split(sys.stdin.readline())[0])
14     sys.stderr.write("java -jar %s %s 2>/dev/null\n" %(config.al_jar, anzahl))
15     r,w=popen2.popen2("java -jar %s %s 2>/dev/null" %(config.al_jar, anzahl))
16     eingabe = sys.stdin.read()    # 2do: matrizenweise lesen und schreiben
17     w.write(eingabe)              # oder besser threaded oder select IO
18     w.close()
19     ausgabe = r.read()
20     r.close()
21     sys.stdout.write(ausgabe)
22
23 if __name__ == '__main__':
24     main()

```

M.3.10 alvis.py

```

1  #!/usr/bin/python
2
3  import os
4  import string
5  import sys
6  import types
7
8  from DatabaseAPI import dokumentdb, db, regiondb, config
9  from Region      import satz, dokument, segment, absatz
10
11 dbconn = db.Db()
12 regiondb.setDefaultDbconn(dbconn)
13
14 def main():
15     global dbconn
16     if len(sys.argv) == 2:
17         align = open(sys.argv[1], "r")
18     else:
19         sys.stderr.write("Benutzung: %s <.align Datei>\n" %sys.argv[0])
20         align = None
21     if align:
22         zeile = align.readline()
23         if zeile[:14] == "Dokument-DB-ID":
24             id = int(zeile[14:])
25             dok1 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["de"])
26             dok2 = dokumentdb.DokumentVonDb(id, dbconn, config.lang2id["en"])
27             print "Dokument-DB-ID", id
28         elif zeile[:14] == "Dokument-Datei":
29             name = string.strip(zeile[15:])
30             dok1 = dokument.DokumentVonDatei(name+".de.tag")
31             dok2 = dokument.DokumentVonDatei(name+".en.tag")
32             print "Dokument-Datei", name
33         else:
34             raise Exception
35         zeile = align.readline()
36         if string.lower(zeile) == "nach satz\n":
37             l1 = dok1.getRegionenMit(satz.Satz)
38             l2 = dok2.getRegionenMit(satz.Satz)
39             segclass = segment.Segment
40         elif string.lower(zeile) == "nach absatz\n":
41             l1 = dok1.getRegionenMit(absatz.Absatz)
42             l2 = dok2.getRegionenMit(absatz.Absatz)
43             segclass = dokument.Dokument

```

```

44     else:
45         raise Exception
46     zeile = align.readline()
47     while zeile != "":
48         zeilenliste = string.split(zeile, "\t") # Zeile am Tab trennen
49         segNr = eval(zeilenliste[0])
50         slice1 = eval(zeilenliste[1])
51         slice2 = eval(zeilenliste[2])
52         seg1 = segclass(l1[slice1[0]:slice1[-1]+1])
53         seg2 = segclass(l2[slice2[0]:slice2[-1]+1])
54         print "--- Segment Nr.", segNr, "Alignment", slice1, "-", slice2, "---"
55         print seg1.getText()
56         print seg2.getText()
57         print
58         zeile = align.readline()
59     align.close()
60     dbconn.close()
61
62 if __name__ == '__main__':
63     main()

```

M.3.11 demo-a2.py

```

1  #!/usr/bin/env python
2
3  import GDK
4  import libglade
5  from gtk          import *
6  from Region      import region, dokument, segment, korpus, absatz
7  from DatabaseAPI import korpusdb, db
8  #import abstand
9
10 class Korpusbaumknoten(GtkTreeItem): #{
11
12     def __init__(self, reg_de, reg_en, statusbar, matrix, sid = None):
13         name = 'reg_de' + ' / ' + 'reg_en'
14         GtkTreeItem.__init__(self, name)
15         self.name = name
16         self.status = statusbar
17         self.reg_de = reg_de
18         self.reg_en = reg_en
19         if not sid:
20             self.sid = statusbar.get_context_id('Korpus')
21         else:
22             self.sid = sid
23         self.matrix = matrix
24         self.exp_id = self.connect("expand", self.sig_expand)
25         self.connect("select", self.sig_select)
26
27     def is_leaf(self):
28         return isinstance(self.reg_de, segment.Segment) \
29             or isinstance(self.reg_en, segment.Segment)
30
31     def init_subtree(self):
32         if not self.is_leaf():
33             self.subtree = GtkTree()
34             self.subtree.set_selection_mode(SELECTION_BROWSE)
35             self.set_subtree(self.subtree)
36             self.subtree.show()
37
38     def sig_select(self, data):

```

```

39     if self.status:
40         self.status.push(self.sid, self.name)
41     if self.matrix:
42         self.matrix.setRegionen(self.reg_de, self.reg_en)
43
44     def sig_expand(self, _t):
45         anzahl = len(self.reg_de)
46         if anzahl > len(self.reg_en):
47             anzahl = len(self.reg_en)
48         for i in range(0, anzahl):
49             item = Korpusbaumknoten(self.reg_de[i],
50                                     self.reg_en[i],
51                                     self.status, self.matrix, self.sid)
52             self.subtree.append(item)
53             item.init_subtree()
54             item.show()
55         self.disconnect(self.exp_id)
56 #}
57
58 class Abstandsmatrix: #{
59
60     def __init__(self, drawingarea, window):
61         self.drawingarea = drawingarea
62         self.reg_de = region.Regionen([]) # leere Regionen
63         self.reg_en = region.Regionen([])
64         self.cellwidth = 14
65         self.cellheight = 14
66         self.width = 10
67         self.height = 10
68         self.xoff = 9
69         self.yoff = 9
70         self.window = window
71         #self.abstand = abstand.AbstandChurchAndGale()
72         #self.abstand = abstand.AbstandTest()
73
74     def setRegionen(self, reg_de, reg_en):
75         self.reg_de = reg_de
76         self.reg_en = reg_en
77         self.draw()
78
79     def draw(self):
80         self.drawingarea.show_now()
81
82     def draw_old(self):
83         reg_de = self.reg_de.getRegionenMit(segment.Segment)
84         reg_en = self.reg_en.getRegionenMit(segment.Segment)
85         x = len(reg_de)
86         y = len(reg_de)
87         if x > 0:
88             x = 2 * self.xoff + self.cellwidth * (x-1)
89         if y > 0:
90             y = 2 * self.yoff + self.cellheight * (y-1)
91         self.drawingarea.size(x, y)
92         context = self.drawingarea.get_window().new_gc()
93         w = self.cellwidth
94         h = self.cellheight
95         for x in range(0, len(reg_de)):
96             for y in range(0, len(reg_en)):
97                 #color = GdkColor(x+8*y,2*y+16*x,13*x+15*y,2)
98                 #context.foreground = color
99                 abstand = self.abstand.zwischen(reg_de[x], reg_en[y])

```

```

100     print abstand, type(abstand)
101     width  = 1 + self.width * abstand
102     height = 1 + self.height * abstand
103     self.drawingarea.draw_rectangle(context, 1,
104     self.xoff + w * x - width/2, self.yoff + h * y - height/2,
105     width, height)
106     # Idee: Abstandsmass = Groesse des Rechtecks
107     self.drawingarea.show_now()
108 #}
109
110 class AbstandsmassDemoWindow(GtkWindow): #{
111
112     def __init__(self, korpus_de, korpus_en):
113         self.widgets = libglade.GladeXML('demo-a2.glade')
114         zeichenflaeche = self.widgets.get_widget('drawingareaMatrix')
115         fenster = self.widgets.get_widget('window1')
116         matrix = Abstandsmatrix(zeichenflaeche, fenster)
117         korpusbaum = self.widgets.get_widget('treeKorpus')
118         statuszeile = self.widgets.get_widget('statusbar1')
119         hauptknoten = Korpusbaumknoten(korpus_de, korpus_en, statuszeile, matrix)
120         korpusbaum.append(hauptknoten)
121         hauptknoten.init_subtree()
122         hauptknoten.show()
123
124     def connect(self, signal, function):
125         fenster = self.widgets.get_widget('window1')
126         fenster.signal_connect(signal, function)
127
128 #}
129
130 def main():
131     #de = korpus.KorpusVonDateiliste(
132     #    ["../wb/wbtag/wb2/sk50de.tag", "../wb/wbtag/wb2/sk500de.tag"])
133     #en = korpus.KorpusVonDateiliste(
134     #    ["../wb/wbtag/wb2/sk50en.tag", "../wb/wbtag/wb2/sk500en.tag"])
135     dbconn = db.Db()
136     de = korpusdb.KorpusVonDb(dbconn, "1")
137     en = korpusdb.KorpusVonDb(dbconn, "2")
138     win = AbstandsmassDemoWindow(de, en)
139     win.connect("destroy", mainquit)
140     win.connect("delete_event", mainquit)
141     mainloop()
142     dbconn.close()
143
144 if __name__ == '__main__':
145     main()
146

```

M.3.12 demo-abstand.py

```

1  #!/usr/bin/python
2
3  libgladetest = 0
4
5  if libgladetest:
6      import libglade
7
8  import popen2
9  import string
10 import sys
11

```

```

12 import tagger
13 import wb
14 import abstand
15 from DatabaseAPI import config, db, regiondb
16 from Region import absatz
17
18 dbconn = db.Db()
19 regiondb.setDefaultDbconn(dbconn)
20
21 ab = { "C&G" : abstand.AbstandChurchAndGale(),
22       "CG" : abstand.AbstandChurchAndGale(),
23       "EINS" : abstand.Abstand(1.0),
24       "TEST" : abstand.AbstandTest(),
25       "NULL" : abstand.Abstand(0.0),
26       "WB" : abstand.AbstandWB(),
27       "WB2" : abstand.AbstandWB_2(),
28       "WB2N" : abstand.AbstandWB_2("de", "en", 1), # nur Nomen
29       "TG" : abstand.AbstandTG(),
30       "TG2" : abstand.AbstandTG_2(),
31     }
32
33 wahl = "WB2"
34
35 sys.stdout.write(
36     """Bitte geben Sie zeilenweise abwechselnd deutschen und englischen Text ein.
37     Mit # kann das Abstandsmass gewechselt werden. ! wendet es auf das letzte
38     Sprachpaar an.
39     Mit +/- kann Abstandinfo ein- bzw. ausgeschaltet werden.
40     """)
41
42 sys.stdout.write("Deutsch:\n")
43 zeile = sys.stdin.readline()
44 de = 1
45 srcseg = None
46 dstseg = None
47 while zeile != "": #{
48     if len(zeile) > 0 and zeile[0] == "#": #{
49         zeile = zeile[1:sys.maxint]
50         zeile = string.strip(zeile)
51         if zeile == "":
52             sys.stdout.write("Aktuelles Abstandsmass %s: %s\n" % ('wahl', 'ab[wahl]'))
53             sys.stdout.write("Moegliche Abstandsmasse: " + 'ab.keys()' + "\n")
54         elif ab.has_key(zeile):
55             wahl = zeile
56             sys.stdout.write("Aktuelles Abstandsmass %s: %s\n" % ('wahl', 'ab[wahl]'))
57         else:
58             sys.stdout.write("Abstandsmass " + 'zeile' + " unbekannt\n")
59     #}
60     elif len(zeile) > 0 and zeile[0] == "+": #{
61         for key in ab.keys():
62             ab[key].setAiout(sys.stdout)
63         sys.stdout.write("Abstandinfo eingeschaltet.\n")
64     #}
65     elif len(zeile) > 0 and zeile[0] == "-": #{
66         for key in ab.keys():
67             ab[key].setAiout(abstand.NullFile())
68         sys.stdout.write("Abstandinfo ausgeschaltet.\n")
69     #}
70     elif len(zeile) > 0 and zeile[0] == "!": #{
71         if srcseg and dstseg:
72             wert = ab[wahl].zwischen(srcseg, dstseg)

```



```

73     sys.stdout.write("Abstand: %s\n" %wert)
74 else:
75     sys.stdout.write("Es wurde weder Quell- noch Zielsegment eingegeben.\n")
76 #}
77 else: #{
78     if de: #{
79         sys.stdout.write("Tagging...\n")
80         srcseg = tagger.tagline(zeile, "de")
81         de = 0
82     #}
83 else: #{
84     sys.stdout.write("Tagging...\n")
85     dstseg = tagger.tagline(zeile, "en")
86     wert = ab[wahl].zwischen(srcseg, dstseg)
87     sys.stdout.write("Abstand:%s\n\n" %wert)
88     de = 1
89 #}
90 #}
91 if de:
92     sys.stdout.write("Bitte deutschen Text eingeben:\n")
93 else:
94     sys.stdout.write("Bitte englischen Text eingeben:\n")
95 zeile = sys.stdin.readline()
96 #}

```

M.3.13 demo-dok.py

```

1  #!/usr/bin/python
2
3  from Region import absatz
4  from Region import atom
5  from Region import dokument
6  from Region import satz
7  from Region import segment
8
9  print "Dokument de-test1.del:"
10 g = dokument.DokumentVonDatei("de-test1.del")
11 print "Text:", g
12 print "Struktur:", `g`
13 for klasse in [absatz.Absatz, segment.Segment, satz.Satz, atom.Atom]:
14     r = g.getRegionenMit(klasse)
15     print len(r), "Regionen zur Klasse", `klasse`
16     for i in range(0, len(r)):
17         print ("["+i+"]": " + `r[i]`)
18     print "Wortliste:", r.getWoerter()
19     print "Tagliste:", r.getTags()
20     print "Lemmaliste:", r.getLemmata()

```

M.3.14 dokvis.py

```

1  #!/usr/bin/python
2
3  import os
4  import string
5  import sys
6  import types
7
8  from DatabaseAPI import dokumentdb, db, regiondb, config
9  from Region import satz, dokument, segment, absatz, korpus
10
11 dbconn = db.Db()

```

```

12 regiondb.setDefaultDbconn(dbconn)
13
14 def main():
15     global dbconn
16     if len(sys.argv) >= 2 and sys.argv[1][:2] == "-r":
17         reg = sys.argv[1][2:]
18         del sys.argv[1]
19         if not reg:
20             reg = sys.argv[1]
21             del sys.argv[1]
22     else:
23         reg = "satz"
24     if len(sys.argv) == 2:
25         dok = dokument.DokumentVonDatei(sys.argv[1])
26     elif len(sys.argv) == 3:
27         dbid = int(sys.argv[1])
28         lang = config.lang2id[sys.argv[2]]
29         dok = dokumentdb.DokumentVonDb(dbid, dbconn, lang)
30     else:
31         sys.stderr.write("Benutzung: %s [<Optionen>] <getaggtes Dokument>\n" %sys.argv[0])
32         sys.stderr.write("oder      : %s [<Optionen>] <db-id> <sprache>\n" %sys.argv[0])
33         sys.stderr.write("Optionen : -r Regionentyp  Werte: satz, segment, absatz\n")
34         dok = None
35     if dok:
36         if reg == "satz":
37             ll = dok.getRegionenMit(satz.Satz)
38         elif reg == "segment":
39             ll = dok.getRegionenMit(segment.Segment)
40         elif reg == "absatz":
41             ll = dok.getRegionenMit(absatz.Absatz)
42         elif reg == "dokument":
43             ll = korpus.Korpus([dok])
44         else:
45             raise Exception
46         for index in range(len(ll)):
47             print ("--- %s Nr." %reg), index, "---"
48             print ll[index].getText()
49             print
50     dbconn.close()
51
52 if __name__ == '__main__':
53     main()

```

M.3.15 eingabe.py

Siehe auch Seite 233 (Schnittstelle) und Seite 231 (Dateiformat).

```

1  #!/usr/bin/python
2
3  # Dieses Modul stellt eine Liste von zu bearbeitenden Dokumentpaaren
4  # bereit, die aus Kommandozeilenparametern gewonnen werden.
5
6  import sys
7  import StringIO
8  from Region import korpus, alignment
9
10 class Liste(korpus.KorpusMitSprache, alignment.AlignedRegion): #{
11
12     def __init__(self, verbose = 0):
13         self.setVerbose(verbose)
14         dokumentliste = []

```

```

15     sprache         = "undefiniert"
16     korpus.KorpusMitSprache.__init__(self, dokumentliste, sprache)
17     self.p_initOptions()
18     self.closeOnExit = []
19     if not regiondb.defaultDbconn:
20         dbconn = db.Db()
21         regiondb.setDefaultDbconn(dbconn)
22         self.closeOnExit.append(dbconn)
23
24     def close(self):
25         for obj in self.closeOnExit:
26             obj.close()
27         self.closeOnExit = []
28
29     def setVerbose(self, verbose):
30         self.verbose = verbose
31
32     def alignedRegion(self):
33         liste = []
34         for i in self:
35             liste.append(i.alignedRegion())
36         return korpus.KorpusMitSprache(liste, self.p_zs) # 2do: auch aligned
37
38     def getQuellsprache(self):
39         return self.p_qs
40
41     def getZielsprache(self):
42         return self.p_zs
43
44     def p_initOptions(self):
45         self.options_l = {
46             'c' : "string    use string as input line",
47             'd' : "language  set destination language",
48             'i' : "filename  read all input from file",
49             'j' : "filename  read dlang-input from file",
50             's' : "language  set source language",
51         }
52         self.options_s = {
53             '.' : "        dump current '-c' buffer",
54             'l' : "        input is a list of input filenames",
55             't' : "        tag input and treat lines as paragraphs",
56         }
57         self.p_finl     = None
58         self.p_fin2     = None
59         self.p_fname_qs = None
60         self.p_fname_zs = None
61         self.p_islist   = 0
62         self.p_tag      = 0
63         self.p_qs       = "de"
64         self.p_zs       = "en"
65         self.p_cmdinput = StringIO.StringIO()
66
67     def eatOptions(self, argv):
68         state = 0
69         error = 0
70         verbose = self.verbose
71         index = 1
72         laenge = len(argv)
73         while index < len(argv): #{
74             arg = argv[index]
75             if not state:

```

```

76         for i in self.options_l.keys():
77             if arg[:2] == "--+i:
78                 state = i
79                 if arg[2:] == "":
80                     del argv[index]
81                 else:
82                     argv[index] = arg[2:]
83                 break
84         for i in self.options_s.keys():
85             if arg[:2] == "--+i:
86                 state = i
87                 if arg[2:] != "":
88                     if verbose:
89                         sys.stderr.write("bad argument %s\n"%'arg')
90                     error = 1
91                 break
92         if not state:
93             break;          # leave eatOptions()
94         continue
95     if state == '.':
96         if verbose:
97             sys.stderr.write("dumping StringIO...\n")
98             cmdinput.seek(0)
99             sys.stdout.write(self.p_cmdinput.read())
100     elif state == 'i':
101         if arg == "-":
102             self.p_fin1 = sys.stdin
103             if verbose:
104                 sys.stderr.write("input1&2 set to stdin\n")
105         else:
106             self.p_fname_qs = arg
107             self.p_fin1 = open(arg, "r")
108             if verbose:
109                 sys.stderr.write("input1&2 set to file %s\n" %arg)
110             self.p_fin2 = self.p_fin1
111     elif state == 'c':
112         self.p_cmdinput.write(arg+'\n')
113         if verbose:
114             sys.stderr.write("StringIO.write(%s)\n"%'arg+'\n')
115     elif state == 's':
116         self.p_qs = arg
117         if verbose:
118             sys.stderr.write("source language = %s\n" %arg)
119     elif state == 't':
120         self.p_tag = 1
121         if verbose:
122             sys.stderr.write("tag = 1\n")
123     elif state == 'l':
124         self.p_islist = 1
125         if verbose:
126             sys.stderr.write("islist = 1\n")
127     elif state == 'd':
128         self.p_zs = arg
129         if verbose:
130             sys.stderr.write("destination language = %s\n" %arg)
131     elif state == 'j':
132         if arg == "-":
133             self.p_fin2 = sys.stdin
134             if verbose:
135                 sys.stderr.write("input2 set to stdin\n")
136     else:

```

```

137         self.p_fname_zs = arg
138         self.p_fin2 = open(arg, "r")
139         if verbose:
140             sys.stderr.write("input2 set to file %s\n" %arg)
141     else:
142         raise NotImplementedError
143     del argv[index]
144     state = 0
145 #}
146 return laenge - len(argv)
147
148 def finishOptions(self):
149     verbose = self.verbose
150     if self.p_cmdinput.tell() != 0:
151         self.p_cmdinput.seek(0)
152         fin1 = self.p_cmdinput
153         fin2 = fin1
154         if verbose:
155             sys.stderr.write("input from '-c' buffer\n")
156         fname_qs = None
157         fname_zs = None
158     else:
159         fname_qs = self.p_fname_qs
160         fname_zs = self.p_fname_zs
161         fin1 = self.p_fin1
162         fin2 = self.p_fin2
163     # create list of DokumentPaar objects according to islist and tag
164     if verbose and fin1 == sys.stdin:
165         sys.stderr.write("expecting input from stdin\n")
166     dokumentPaare = []
167     qs = self.p_qs
168     zs = self.p_zs
169     if fin1 and self.p_islist:
170         fnamedic = {}
171         fname_qs = fin1.readline()
172         fname_zs = fin2.readline()
173         while fname_qs and fname_zs:
174             fname_qs = fname_qs[:-1]      # Newline abschneiden
175             fname_zs = fname_zs[:-1]
176             if not self.p_tag:
177                 dp = dokument.DokumentPaarVonDateien(fname_qs, fname_zs,
178                                                         qs, zs)
179             else:
180                 tfin1 = open(fname_qs, "r")
181                 tfin2 = open(fname_zs, "r")
182                 dp = dokument.DokumentPaarVonRohtexte(tfin1, tfin2, qs, zs)
183                 tfin1.close()
184                 tfin2.close()
185             dokumentPaare.append(dp)
186             fname_qs = fin1.readline()
187             fname_zs = fin2.readline()
188     elif fin1:
189         if fname_qs and fname_zs:
190             dp = dokument.DokumentPaarVonDateien(fname_qs, fname_zs, qs,zs)
191         elif not self.p_tag:
192             dp = dokument.DokumentPaarVonStreams(fin1, fin2, qs, zs)
193         else:
194             dp = dokument.DokumentPaarVonRohtexte(fin1, fin2, qs, zs)
195         dokumentPaare.append(dp)
196     if fin1:
197         fin1.close()

```

```

198         if fin1 != fin2:
199             fin2.close()
200     self.p_setDokumentliste(dokumentPaare)
201     self.p_setSprache(qs)
202
203     def help(self, file):
204         keys = self.options_l.keys() + self.options_s.keys()
205         keys.sort()
206         for key in keys:
207             if self.options_l.has_key(key):
208                 file.write("        -%s %s\n" %(key, self.options_l[key]))
209             elif self.options_s.has_key(key):
210                 file.write("        -%s %s\n" %(key, self.options_s[key]))
211             else:
212                 raise ValueError
213
214 #}
215
216 from DatabaseAPI import db, regiondb
217
218 #
219 # getRegion() liefert RegionenMit-Objekt zum als String gegebenen Regionentyp
220 #
221
222 def getRegionen(dok, reg):
223     if reg == "satz":
224         ll = dok.getRegionenMit(satz.Satz)
225     elif reg == "segment":
226         ll = dok.getRegionenMit(segment.Segment)
227     elif reg == "absatz":
228         ll = dok.getRegionenMit(absatz.Absatz)
229     elif reg == "dokument":
230         ll = dok.getRegionenMit(dokument.Dokument)
231     elif reg == "korpus":
232         ll = dok.getRegionenMit(korpus.Korpus)
233     else:
234         raise Exception
235     return ll
236
237 #####
238 # Test-Skript
239
240 from Region import korpus, dokument, absatz, segment, satz
241
242 def main():
243     nebeneinander = 0
244     verbose = 0
245     hilfe = 0
246     reg = "satz"
247     dok = Liste(verbose)
248     while len(sys.argv) > 1: #{
249         #if verbose:
250         #    sys.stderr.write("args left: %s\n" % `sys.argv[1:]`)
251         if sys.argv[1] == "-h":
252             hilfe = 1
253             break
254         elif sys.argv[1] == "-v":
255             verbose = 1
256             dok.setVerbose(verbose)
257             del sys.argv[1]
258         elif sys.argv[1] == "-n":

```

```

259     nebeneinander = 1
260     del sys.argv[1]
261 elif sys.argv[1][:2] == "-r":
262     reg = sys.argv[1][2:]
263     del sys.argv[1]
264     if not reg:
265         try:
266             reg = sys.argv[1]
267             del sys.argv[1]
268         except:
269             hilfe = 1
270             break
271 else:
272     eaten = dok.eatOptions(sys.argv)
273     if not eaten:
274         hilfe = 1
275         break
276     if verbose:
277         sys.stderr.write("eingabe.Liste has eaten %s args\n" %eaten)
278 if hilfe:
279     if verbose:
280         sys.stderr.write("args left: %s\n" %`sys.argv[1:]`)
281     sys.stderr.write("Usage: %s <options>\n" %sys.argv[0])
282     sys.stderr.write("options:  -h          prints this help\n")
283     sys.stderr.write("          -n          alternate docs in output\n")
284     sys.stderr.write("          -r regtype   soft regions: one of\n")
285     for i in ["satz", "segment", "absatz", "dokument"]:
286         sys.stderr.write(27*' ' + "%s\n" %i)
287     sys.stderr.write("          -v          verbose\n")
288     dok.help(sys.stderr)
289     dok.close()
290     sys.exit(0)
291 if verbose:
292     sys.stderr.write("reading input...\n")
293 dok.finishOptions()
294 if verbose:
295     sys.stderr.write("region type = %s\n" %reg)
296 l1 = getRegionen(dok, reg)
297 l2 = getRegionen(dok.alignedRegion(), reg)
298 if verbose:
299     sys.stderr.write("number of regions: %s\n" %`(len(l1),len(l2))`)
300 if nebeneinander:
301     for index in range(max(len(l1), len(l2))):
302         print ("--- %s Nr." %reg), index, "---"
303         if l1.has_key(index):
304             print l1[index].getText()
305         if l2.has_key(index):
306             print l2[index].getText()
307         print
308 else:
309     for (header, l) in [("Quellsprache", l1), ("Zielsprache", l2)]:
310         print "### %s ###" %header
311         print
312         for index in range(len(l)):
313             print ("--- %s Nr." %reg), index, "---"
314             print l[index].getText()
315             print
316 dok.close()
317
318 if __name__ == '__main__':
319     main()

```

M.3.16 getalign.py

```

1  #!/usr/bin/python
2
3  import Abstand
4  import os
5  import sys
6  import types
7
8  from DatabaseAPI import dokumentdb, db, regiondb, config
9  from Region      import satz, dokument, segment
10
11 dbconn = db.Db()
12 regiondb.setDefaultDbconn(dbconn)
13
14 blockgroesse = 1
15 if blockgroesse != 1:
16     print "Derzeit nur Blockgroesse 1 unterstuetzt."
17     sys.exit(1)
18
19 def usage():
20     print "Usage: getalign.py -d <DB-IDs> | -s <basename>"
21     sys.exit()
22
23 def main():
24     global dbconn
25     if len(sys.argv) < 2:
26         usage()
27     if sys.argv[1] == "-d":
28         liste = map(lambda x: int(x), sys.argv[2:])
29     elif sys.argv[1] == "-s":
30         liste = sys.argv[2:]
31     else:
32         usage()
33     for name in liste:
34         if type(name) == types.StringType:
35             fname = name
36         elif type(name) == types.IntType:
37             fname = 'db_'+'name'
38         else:
39             raise TypeError
40         align = open(os.path.basename(fname)+".align", "w")
41         if type(name) == types.StringType:
42             dok1 = dokument.DokumentVonDatei(name+".de.tag.al")
43             dok2 = dokument.DokumentVonDatei(name+".en.tag.al")
44             align.write("Dokument-Datei "+name+"\n")
45         elif type(name) == types.IntType:
46             dok1 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["de"])
47             dok2 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["en"])
48             align.write("Dokument-DB-ID "+name+"\n")
49         else:
50             raise TypeError
51         l1 = dok1.getRegionenMit(segment.Segment)
52         l2 = dok2.getRegionenMit(segment.Segment)
53         align.write("nach Satz\n")
54         s1 = 0
55         s2 = 0
56         laenge = len(l1)
57         if len(l2) < laenge:
58             laenge = len(l2)
59         sys.stderr.write("Warnung: " + name + " Anzahl Seg. verschieden\n")

```



```

60     for i in range(0,laenge):
61         s1n = s1+len(l1[i])
62         s2n = s2+len(l2[i])
63         align.write('i'           + \
64                    '\t' +         'range(s1,s1n)' + \
65                    '\t' +         'range(s2,s2n)' + "\n")
66         s1 = s1n
67         s2 = s2n
68     align.close()
69     dbconn.close()
70
71 if __name__ == '__main__':
72     main()

```

M.3.17 matrix.py

```

1  #!/usr/bin/python
2
3  import abstand
4  import os
5  import sys
6  import types
7
8  from DatabaseAPI import dokumentdb, db, regiondb, config
9  from Region       import satz, dokument, segment
10
11 dbconn = db.Db()
12 regiondb.setDefaultDbconn(dbconn)
13
14 blockgroesse = 1
15 if blockgroesse != 1:
16     sys.stderr.write("Derzeit nur Bloeckgroesse 1 unterstuetzt."+'\n')
17     sys.exit(1)
18
19 def main():
20     global dbconn
21     jobs={}
22     jobs["default"] = {}
23     #jobs["default"]["abstand"] = abstand.Abstand()
24     #jobs["default"]["abstand"] = abstand.AbstandTest()
25     #jobs["default"]["abstand"] = abstand.AbstandChurchAndGale()
26     #jobs["default"]["abstand"] = abstand.AbstandTG()
27     #jobs["default"]["abstand"] = abstand.AbstandWB()
28     jobs["default"]["abstand"] = abstand.AbstandWB_2()
29     #jobs["default"]["abstand"] = abstand.AbstandWB_2("de","en",1) # nurNomen
30     for mass in jobs.keys():
31         jobs[mass]["file"] = sys.stdout
32     for name in sys.argv[1:]:
33         if type(name) == types.StringType:
34             fname = name
35         elif type(name) == types.IntType:
36             fname = 'db_'+name
37         else:
38             raise TypeError
39         if type(name) == types.StringType:
40             dok1 = dokument.DokumentVonDatei(name+".de.tag")
41             dok2 = dokument.DokumentVonDatei(name+".en.tag")
42             for mass in jobs.keys():
43                 jobs[mass]["file"].write("Dokument-Datei "+name+"\n")
44         elif type(name) == types.IntType:
45             dok1 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["de"])

```

```

46         dok2 = dokumentdb.DokumentVonDb(name, dbconn, config.lang2id["en"])
47         for mass in jobs.keys():
48             jobs[mass]["file"].write("Dokument-DB-ID "+name+"\n")
49     else:
50         raise TypeError
51     l1 = dok1.getRegionenMit(segment.Segment)
52     l2 = dok2.getRegionenMit(segment.Segment)
53     s1 = 0
54     s2 = 0
55     laenge = len(l1)
56     if len(l2) < laenge:
57         laenge = len(l2)
58     sys.stderr.write("Warnung: " + name + " Anzahl Seg. verschieden\n")
59     for i in range(0,laenge):
60         s1n = s1+len(l1[i])
61         s2n = s2+len(l2[i])
62         s1 = s1n
63         s2 = s2n
64     l1 = dok1.getRegionenMit(satz.Satz)
65     l2 = dok2.getRegionenMit(satz.Satz)
66     for job in jobs.values():
67         job["file"].write("nach Satz\n")
68         job["file"].write('\len(l2)-blockgroesse+1' + "\t")
69         job["file"].write('\len(l1)-blockgroesse+1' + "\n")
70     for y in range(0, len(l1)-blockgroesse+1):
71         for x in range(0, len(l2)-blockgroesse+1):
72             # sys.stderr.write('\n')
73             # sys.stderr.write(("--- %s - %s ---" %('y','x'))+'\n')
74             # sys.stderr.write(("DE: %s" %l1[y].getText())+'\n')
75             # sys.stderr.write(("EN: %s" %l2[x].getText())+'\n')
76             # sys.stderr.write('\n')
77             abstaende=[]
78             for job in jobs.values():
79                 a = job["abstand"].zwischen(l1[y], l2[x])
80                 job["file"].write('a' + "\n")
81                 abstaende.append(a)
82             # sys.stderr.write("AB: %s" %'abstaende')
83             # sys.stderr.write('name' + " erledigt.\n")
84     jobs["default"]["abstand"].close()
85     dbconn.close()
86
87 def dateiliste():
88     return [
89     './korpus/de-news/1997/05/de-news-1997-05-30',
90     403,
91     './korpus/de-news/1997/05/de-news-1997-05-19',
92     './korpus/de-news/1997/05/de-news-1997-05-03',
93     './korpus/de-news/1997/05/de-news-1997-05-11',
94     './korpus/de-news/1997/05/de-news-1997-05-06',
95     ]
96
97 dummyliste = [
98     './korpus/de-news/1997/05/de-news-1997-05-01',
99     './korpus/de-news/1997/05/de-news-1997-05-02',
100    './korpus/de-news/1997/05/de-news-1997-05-03',
101    './korpus/de-news/1997/05/de-news-1997-05-04',
102    './korpus/de-news/1997/05/de-news-1997-05-05',
103    './korpus/de-news/1997/05/de-news-1997-05-06',
104    './korpus/de-news/1997/05/de-news-1997-05-07',
105    './korpus/de-news/1997/05/de-news-1997-05-08',
106    './korpus/de-news/1997/05/de-news-1997-05-09',

```

```

107 './korpus/de-news/1997/05/de-news-1997-05-10',
108 './korpus/de-news/1997/05/de-news-1997-05-11',
109 './korpus/de-news/1997/05/de-news-1997-05-12',
110 './korpus/de-news/1997/05/de-news-1997-05-13',
111 './korpus/de-news/1997/05/de-news-1997-05-14',
112 './korpus/de-news/1997/05/de-news-1997-05-15',
113 './korpus/de-news/1997/05/de-news-1997-05-16',
114 './korpus/de-news/1997/05/de-news-1997-05-17',
115 './korpus/de-news/1997/05/de-news-1997-05-19',
116 './korpus/de-news/1997/05/de-news-1997-05-20',
117 './korpus/de-news/1997/05/de-news-1997-05-21',
118 './korpus/de-news/1997/05/de-news-1997-05-22',
119 './korpus/de-news/1997/05/de-news-1997-05-23',
120 './korpus/de-news/1997/05/de-news-1997-05-24',
121 './korpus/de-news/1997/05/de-news-1997-05-26',
122 './korpus/de-news/1997/05/de-news-1997-05-27',
123 './korpus/de-news/1997/05/de-news-1997-05-28',
124 './korpus/de-news/1997/05/de-news-1997-05-30',
125 './korpus/de-news/1997/05/de-news-1997-05-31',
126 ]
127
128 if __name__ == '__main__':
129     main()

```

M.3.18 matrix.sh

```

1 #!/bin/bash
2
3 # matrix.sh erstellt eine fuer ein Dokumentpaar
4 #   1. absatzweise Abstandsmatrizen in $1_$2.abs.mtr
5 #   2. Gesamtabstandsmatrix           in $1_$2.mtr
6 #   3. Fehler- und Diagnoseausgabe   in temp.out
7 # und fuer WB2 zusaetzlich
8 #   4. absatzweise Abstandinfo       in $I.abs.ai
9 #   5. Gesamtabstandinfo             in $I.abstandinfo
10
11 # Aufruf: matrix.sh dokumentprefix abstandsmass
12
13 ALIGN='dirname $0'
14
15 test "$2" = WB2 && python "$ALIGN/nmatrix.py" -v -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" -o "$1_
16
17 test "$2" != WB2 && python "$ALIGN/nmatrix.py" -v -m "$2" -r satz -R absatz -i "$1.de.tag" -j "$1.en.tag" -o "$1_

```

M.3.19 mavis.sh

```

1 #!/bin/bash
2
3 #JAVA=/usr/lib/jdk1.3/bin/java
4 JAVA=java
5
6 $JAVA de.denkselbst.mavis.Mavis $1

```

M.3.20 mergemtr.py

```

1 #!/usr/bin/python
2
3
4 import string
5 import sys
6

```

```

7  aain = None # Anstandinfo
8  aiout = None
9  if len(sys.argv) > 1:
10     aain = open(sys.argv[1], "r")
11  if len(sys.argv) > 2:
12     aiout = open(sys.argv[2], "w")
13
14  state = "header"
15  breiten = []
16  hoehen = []
17  matrizen = []
18  abstandinfos = []
19
20  def fehler(ueberschrift):
21     sys.stderr.write("%s\n" %ueberschrift)
22     sys.stderr.write("aktuelle Zeile: zeile = %s\n" %'zeile')
23     sys.stderr.write("bisher gelesen: daten = %s\n" %'daten')
24     sys.stderr.write("bisher gelesen: len = %s\n" %'len(daten)')
25     sys.stderr.write("notwendig: breite*hoehe = %s\n" %'breite*hoehe')
26     sys.exit(1)
27
28  header_ausgegeben = 0
29  zeile = sys.stdin.readline()
30  while zeile != "":
31     if state == "header":
32         if zeile[:len("Absatz ")] == "Absatz ":
33             if not header_ausgegeben:
34                 sys.stdout.write(zeile[string.find(zeile, " von")+5:])
35                 state = "header2"
36             elif zeile[-len(" jobs follow\n"):] == " jobs follow\n":
37                 state = "header"
38             else:
39                 sys.stderr.write("Quelleninfo %s nicht verstanden\n" %'zeile')
40                 raise Exception
41         elif state == "header2":
42             if string.lower(zeile) == "nach satz\n":
43                 if not header_ausgegeben:
44                     sys.stdout.write(zeile)
45                     header_ausgegeben = 1
46                 state = "header3"
47             else:
48                 sys.stderr.write("Softregiontyp %s nicht verstanden\n" %'zeile')
49                 raise Exception
50         elif state == "header3":
51             liste = string.split(zeile, '\t')
52             if len(liste) == 2:
53                 try:
54                     breite = int(liste[0])
55                     hoehe = int(liste[1])
56                     daten = []
57                     aidaten = []
58                     state = "daten"
59                     #sys.stderr.write("Teilmatrix %s\n" %'(breite,hoehe)')
60                     if len(daten) == breite * hoehe:
61                         breiten.append(breite)
62                         hoehen.append(hoehe)
63                         matrizen.append(daten)
64                         abstandinfos.append(aidaten)
65                         state = "header"
66             except ValueError:
67                 pass

```

```

68 elif state == "daten":
69     liste = string.split(zeile)
70     try:
71         for datum in liste:
72             daten.append(float(datum))
73             aidatum = []
74             if aiin:
75                 aizeile = aiin.readline()
76                 while aizeile != "<SATZ>\n":
77                     aidatum.append(string.rstrip(aizeile))
78                     if aizeile == "":
79                         fehler("EOF in abstandinfos zu frueh")
80                         raise Exception      # EOF zu frueh
81                     aizeile = aiin.readline()
82                     aidatum.append("")      # fuer abschliessendes Newline
83             aidaten.append(string.join(aidatum, '\n'))
84     except ValueError:
85         fehler("Fehlerhafte Eingabe")
86     if len(daten) >= breite * hoehe:
87         breiten.append(breite)
88         hoehen.append(hoehe)
89         matrizen.append(daten)
90         abstandinfos.append(aidaten)
91         state = "header"
92     zeile = sys.stdin.readline()
93     breite = 0
94     for i in breiten:
95         breite = breite + i
96     hoehe = 0
97     for i in hoehen:
98         hoehe = hoehe + i
99     sys.stdout.write("%s\t%s\n" %(breite, hoehe))
100    if aiout:
101        aiout.write("%s\t%s\n" %(breite, hoehe))
102    prefix = 0
103    suffix = breite
104    for mindex in range(len(matrizen)):
105        mhoehe = hoehen[mindex]
106        mbreite = breiten[mindex]
107        matrix = matrizen[mindex]
108        abstandinfo = abstandinfos[mindex]
109        suffix = suffix - mbreite
110        for my in range(mhoehe):
111            sys.stdout.write(prefix * "1.0\n")
112            if aiout:
113                aiout.write(prefix * "nicht berechnet, da ausserhalb des Absatzalignments\n<SATZ>\n")
114            for mx in range(mbreite):
115                sys.stdout.write("%s\n" %matrix[mx+my*mbreite])
116                if aiout:
117                    aiout.write(abstandinfo[mx+my*mbreite])
118                    aiout.write("<SATZ>\n")
119            sys.stdout.write(suffix * "1.0\n")
120            if aiout:
121                aiout.write(suffix * "nicht berechnet, da ausserhalb des Absatzalignments\n<SATZ>\n")
122        prefix = prefix + mbreite
123
124

```

M.3.21 mtr2mavis.py

```

1  #!/usr/bin/python
2
3  # mtr2mavis.py
4  # passt die Abstandsmatrizen fuer Dateien, die nicht dem Schema
5  # $STAMM.de.tag / $STAMM.en.tag entsprechen, an und legt Links
6  # an, um nach diesem Schema auf die Dateien zugreifen zu koennen.
7
8  import os
9  import string
10 import sys
11
12 if len(sys.argv) <= 1:
13     sys.stderr.write("Benutzung: %s <Liste von .mtr Dateien>\n" %sys.argv[0])
14     sys.exit(1)
15
16 def neueNamen(name1, name2):
17     ldp = min([len(name1), len(name2)])
18     while name1[:ldp] != name2[:ldp]:
19         ldp = ldp - 1
20     lfdnr = 1
21     while 1:
22         stamm = name1[:ldp] + "~" + `lfdnr`
23         name_de = stamm + ".de.tag"
24         name_en = stamm + ".en.tag"
25         name_mtr = stamm + ".mtr"
26         if not os.access(name_de, os.F_OK) \
27            and not os.access(name_en, os.F_OK) \
28            and not os.access(name_mtr, os.F_OK):
29             break
30         lfdnr = lfdnr + 1
31     return (name_de, name_en, name_mtr, stamm)
32
33 for mtrfilename in sys.argv[1:]:
34
35     mtrfile = open(mtrfilename, "r")
36     zeile = mtrfile.readline()
37     if zeile[:len("Dokument-Datei ")] == "Dokument-Datei ":
38         sys.stderr.write("%s bereits im richtigem Format\n" %mtrfilename)
39         mtrfile.close()
40         continue
41     elif zeile[-len(" jobs follow\n"):] == " jobs follow\n":
42         sys.stderr.write("%s erfordert zuerst mergemtr.py \n" %mtrfilename)
43         mtrfile.close()
44         continue
45     elif zeile[:len("Paar Dokument-Datei ")] == "Paar Dokument-Datei ":
46         pos = string.find(zeile, " und Dokument-Datei")
47         name1 = string.strip(zeile[20:pos])
48         name2 = string.strip(zeile[pos+20:])
49         name_de, name_en, name_mtr, stamm = neueNamen(name1, name2)
50         mtrneu = open(name_mtr, "w")
51         mtrneu.write("Dokument-Datei %s\n" %stamm) # Header neu
52         mtrneu.write(mtrfile.read()) # Rest kopieren
53         mtrneu.close()
54         mtrfile.close()
55         os.link(name1, name_de)
56         os.link(name2, name_en)
57         name_ai = name1 + ".abstandinfo"
58         if os.access(name_ai, os.F_OK):
59             os.link(name_ai, stamm + ".abstandinfo")

```

```

60     else:
61         sys.stderr.write("%s nicht behandelt, Header unbekannt\n" %mtrfilename)
62         mtrfile.close()
63

```

M.3.22 nmatrix.py

```

1  #!/usr/bin/python
2
3  # Kommandozeilentool matrix.py
4
5  import math
6  import sys
7
8  import abstand
9  import eingabe
10 from DatabaseAPI import config, db, regiondb
11 from Region      import atom, absatz, region
12
13 def main():
14     dbconn = db.Db()
15     regiondb.setDefaultDbconn(dbconn)
16     ab = { "C&G"   : abstand.AbstandChurchAndGale(),
17           "CG"    : abstand.AbstandChurchAndGale(),
18           "EINS"  : abstand.Abstand(1.01),
19           "TEST"  : abstand.AbstandTest(),
20           "NULL"  : abstand.Abstand(0.0),
21           "WB"    : abstand.AbstandWB(),
22           "WB2"   : abstand.AbstandWB_2(),
23           "WB2N"  : abstand.AbstandWB_2("de","en",1),    # nur Nomen
24           "TG"    : abstand.AbstandTG(),
25           "TG2"   : abstand.AbstandTG_2(),
26     }
27     wahl = "TG"
28     fout = sys.stdout
29     aiout = None
30     verbose = 0
31     text = 0
32     leave_out = 0
33     leave_out_dist_gradient = 0.0
34     leave_out_dist_max = -1.0
35     hilfe = len(sys.argv) == 1
36     reg_soft = "segment"
37     reg_hard = "dokument"
38     korp = eingabe.Liste(verbose)
39     while len(sys.argv) > 1: #{
40         #if verbose:
41             # sys.stderr.write("args left: %s\n" %`sys.argv[1:]`)
42             if sys.argv[1] == "-h":
43                 hilfe = 1
44                 break
45             elif sys.argv[1] == "-u":
46                 text = 1
47                 del sys.argv[1]
48             elif sys.argv[1] == "-v":
49                 verbose = 1
50                 korp.setVerbose(verbose)
51                 del sys.argv[1]
52             elif sys.argv[1][:2] == "-l":
53                 leave_out_dist = sys.argv[1][2:]
54                 del sys.argv[1]

```

```

55     if not leave_out_dist:
56         try:
57             leave_out_dist = sys.argv[1]
58             del sys.argv[1]
59         except:
60             hilfe = 1
61             break
62     try:
63         leave_out_dist = float(leave_out_dist)
64     except ValueError:
65         hilfe = 1
66         break
67     leave_out = 1
68     if verbose:
69         sys.stderr.write("cells more than %s units " %leave_out_dist)
70         sys.stderr.write("from the conectiong line will be left out\n")
71 elif sys.argv[1][:2] == "-L":
72     leave_out_dist_gradient = sys.argv[1][2:]
73     del sys.argv[1]
74     if not leave_out_dist_gradient:
75         try:
76             leave_out_dist_gradient = sys.argv[1]
77             del sys.argv[1]
78         except:
79             hilfe = 1
80             break
81     try:
82         leave_out_dist_gradient = float(leave_out_dist_gradient)
83     except ValueError:
84         hilfe = 1
85         break
86     if verbose:
87         sys.stderr.write("The leave out distance will be increased by ")
88         sys.stderr.write("%s units for each " %leave_out_dist_gradient)
89         sys.stderr.write("unit along the\n connecting line (upper left")
90         sys.stderr.write(" to lower right) up to the middle of each ")
91         sys.stderr.write("hard-region\n")
92 elif sys.argv[1][:2] == "-M":
93     leave_out_dist_max = sys.argv[1][2:]
94     del sys.argv[1]
95     if not leave_out_dist_max:
96         try:
97             leave_out_dist_max = sys.argv[1]
98             del sys.argv[1]
99         except:
100             hilfe = 1
101             break
102     try:
103         leave_out_dist_max = float(leave_out_dist_max)
104     except ValueError:
105         hilfe = 1
106         break
107     if verbose:
108         sys.stderr.write("If -l is used, leave out distance will not ")
109         sys.stderr.write("exceed %s units.\n" %leave_out_dist_max)
110 elif sys.argv[1][:2] == "-r":
111     reg_soft = sys.argv[1][2:]
112     del sys.argv[1]
113     if not reg_soft:
114         try:
115             reg_soft = sys.argv[1]

```



```
116         del sys.argv[1]
117     except:
118         hilfe = 1
119         break
120     if verbose:
121         sys.stderr.write("soft region type = %s\n" %reg_soft)
122 elif sys.argv[1][:2] == "-R":
123     reg_hard = sys.argv[1][2:]
124     del sys.argv[1]
125     if not reg_hard:
126         try:
127             reg_hard = sys.argv[1]
128             del sys.argv[1]
129         except:
130             hilfe = 1
131             break
132     if verbose:
133         sys.stderr.write("hard region type = %s\n" %reg_hard)
134 elif sys.argv[1][:2] == "-m":
135     wahl = sys.argv[1][2:]
136     del sys.argv[1]
137     if not wahl:
138         try:
139             wahl = sys.argv[1]
140             del sys.argv[1]
141         except:
142             hilfe = 1
143             break
144     if verbose:
145         sys.stderr.write("distance measure = %s\n" %wahl)
146 elif sys.argv[1][:2] == "-o":
147     fname = sys.argv[1][2:]
148     del sys.argv[1]
149     if not fname:
150         try:
151             fname = sys.argv[1]
152             del sys.argv[1]
153         except:
154             hilfe = 1
155             break
156     fout = open(fname, "w")
157     if verbose:
158         sys.stderr.write("output to file %s\n" %fname)
159 elif sys.argv[1][:2] == "-O":
160     fname = sys.argv[1][2:]
161     del sys.argv[1]
162     if not fname:
163         try:
164             fname = sys.argv[1]
165             del sys.argv[1]
166         except:
167             hilfe = 1
168             break
169     aiout = open(fname, "w")
170     if verbose:
171         sys.stderr.write("abstandinfo %s\n" %fname)
172 else:
173     eaten = korp.eatOptions(sys.argv)
174     if not eaten:
175         hilfe = 1
176         break
```

```

177         if verbose:
178             sys.stderr.write("eingabe.Liste has eaten %s args\n" %eaten)
179     if hilfe:
180         if verbose:
181             sys.stderr.write("args left: %s\n" %`sys.argv[1:]`)
182         sys.stderr.write("Usage: %s <options>\n" %sys.argv[0])
183         sys.stderr.write("options:  -h          print out this help\n")
184         sys.stderr.write("      -l n          leave out cell which are more than n units away\n")
185         sys.stderr.write("                  from the coneciong line\n")
186         sys.stderr.write("      -L gradient  modifies leave out distance\n")
187         sys.stderr.write("      -M maximum   modifies leave out distance\n")
188         sys.stderr.write("      -m distmeas  choose distance measure:\n")
189         keys = ab.keys()
190         keys.sort()
191         for w in keys:
192             sys.stderr.write(25*' ' + "%s\t%s\n" %(w, `ab[w]`))
193         sys.stderr.write("      -o filename  write output to file\n")
194         sys.stderr.write("      -O filename  write abstandinfo to file\n")
195         sys.stderr.write("      -r regtype   soft regions: one of\n")
196         for i in ["satz", "segment", "absatz", "dokument"]:
197             sys.stderr.write(25*' ' + "%s\n" %i)
198         sys.stderr.write("      -R regtype   hard regions: one of\n")
199         for i in ["segment", "absatz", "dokument", "korpus"]:
200             sys.stderr.write(25*' ' + "%s\n" %i)
201         sys.stderr.write("      -u          output index and text\n")
202         sys.stderr.write("      -v          verbose mode\n")
203         korp.help(sys.stderr)
204         korp.close()
205         for key in ab.keys():
206             ab[key].close()
207         dbconn.close()
208         sys.exit(1)
209     if verbose:
210         sys.stderr.write("distance measure %s = %s\n" %(wahl, `ab[wahl]`))
211     if aiout:
212         for key in ab.keys():
213             ab[key].setAiout(aiout) # Abstandinfo
214     korp.finishOptions()
215     hardregions = eingabe.getRegionen(korp, reg_hard)
216     fout.write("%s jobs follow\n" %len(hardregions))
217     try:
218         dok = hardregions[len(hardregions)-1]
219         alignedDok = dok.alignedRegion()
220     except AttributeError:
221         sys.stderr.write("Error: no aligned hardregion for last hardregion")
222         sys.stderr.write(" %s\n" %`dok`)
223         korp.close()
224         for key in ab.keys():
225             ab[key].close()
226         dbconn.close()
227         sys.exit(1)
228     for hreg_index in range(len(hardregions)):
229         dok = hardregions[hreg_index]
230         fout.write("%s\n" %dok.quelleninfo())
231         fout.write("nach %s\n" %reg_soft)
232         try:
233             alignedDok = dok.alignedRegion()
234         except AttributeError:
235             sys.stderr.write("Error: no aligned hardregion for")
236             sys.stderr.write(" %s\n" %`dok`)
237         korp.close()

```

```

238     for key in ab.keys():
239         ab[key].close()
240     dbconn.close()
241     sys.exit(1)
242 if korp.getQuellsprache() == "de" and korp.getZielsprache() == "en":
243     l1 = eingabe.getRegionen(dok, reg_soft)
244     l2 = eingabe.getRegionen(alignedDok, reg_soft)
245 elif korp.getQuellsprache() == "en" and korp.getZielsprache() == "de":
246     l1 = eingabe.getRegionen(alignedDok, reg_soft)
247     l2 = eingabe.getRegionen(dok, reg_soft)
248 elif not wahl in ["WB"]: # Liste der sprachabhaengigen Abstandsmasse
249     l1 = eingabe.getRegionen(dok, reg_soft) # Reihenfolge egal
250     l2 = eingabe.getRegionen(alignedDok, reg_soft)
251 else:
252     sys.stderr.write("Error: cannot compare %s"%korp.getQuellsprache())
253     sys.stderr.write(" and %s" %korp.getZielsprache())
254     sys.stderr.write(" with %s\n" %`ab[wahl]`)
255     korp.close()
256     for key in ab.keys():
257         ab[key].close()
258     dbconn.close()
259     sys.exit(1)
260 if verbose:
261     sys.stderr.write("number of regions: %s\n" %`(len(l1),len(l2))`)
262 laengel = len(l1)
263 laenge2 = len(l2)
264 if laengel <= 1 or laenge2 <= 1:
265     mab = ab["NULL"]
266 else:
267     mab = ab[wahl]
268 mab2 = mab
269 if leave_out:
270     vec1 = [laenge2 - 1, laengel - 1] # Verbindungslinie ol nach ur
271     vec2 = [laengel - 1, 1 - laenge2] # senkrecht zur Verbindungsl.
272     laenge = math.sqrt(vec2[0]*vec2[0]+vec2[1]*vec2[1])
273     if laenge > 0.01:
274         vec1[0] = vec1[0] / laenge # Vektor normieren
275         vec1[1] = vec1[1] / laenge
276         vec2[0] = vec2[0] / laenge
277         vec2[1] = vec2[1] / laenge
278     else:
279         vec1[0] = 1 / math.sqrt(2) # default fuer 1:1
280         vec1[1] = 1 / math.sqrt(2)
281         vec2[0] = 1 / math.sqrt(2)
282         vec2[1] = -1 / math.sqrt(2)
283     if verbose:
284         sys.stderr.write("orthonormal vector %s\n" %vec2)
285 fout.write("%s\t%s\n" %(laenge2, laengel))
286 for y in range(laengel):
287     for x in range(laenge2):
288         if leave_out:
289             dist_x = x * vec1[0] + y * vec1[1] # Skalarprodukt
290             dist_y = x * vec2[0] + y * vec2[1] # Skalarprodukt
291             dx = min([dist_x, laenge - dist_x])
292             lleave_out_dist = leave_out_dist + \
293                 leave_out_dist_gradient * dx
294             if lleave_out_dist > leave_out_dist_max >= 0.0:
295                 lleave_out_dist = leave_out_dist_max
296             if math.fabs(dist_y) > lleave_out_dist:
297                 mab2 = ab["EINS"]
298         else:

```

```

299         mab2 = mab
300     if text:
301         fout.write("--- %s Nr. %s zu %s---\n" %(reg_soft, y, x))
302         if l1.has_key(y):
303             fout.write(l1[y].getText()+'\n')
304         if l2.has_key(x):
305             fout.write(l2[x].getText()+'\n')
306     wert = mab2.zwischen(l1[y], l2[x])
307     fout.write("%s\n" %`wert`)
308     if text:
309         fout.write('\n')
310     if verbose:
311         sys.stderr.write(`mab.statistic()`+'\n')
312     korp.close()
313     for key in ab.keys():
314         ab[key].close()
315     dbconn.close()
316     if aiout:
317         aiout.close()
318
319 if __name__ == '__main__':
320     main()

```

M.3.23 zeitmessung.py

```

1  #!/usr/bin/python
2
3  # Modul zeitmessung
4
5  # zwei alternative Basisklassen, fuer Klassen, die die Verweildauer
6  # in verschiedenen Funktionen messen wollen
7
8  import sys
9  import time
10 import types
11
12 class ZeitmessungNull:
13
14     def __init__(self):
15         self.zeit_in = {}
16
17     def statistic(self):
18         return self.zeit_in
19
20     def p_begin(self, name):
21         pass
22
23     def p_end(self, name):
24         pass
25
26 class Zeitmessung(ZeitmessungNull):
27
28     def __init__(self):
29         ZeitmessungNull.__init__(self)
30
31     def p_begin(self, name):
32         if self.zeit_in.has_key(name):
33             bis_jetzt = self.zeit_in[name]
34         else:
35             bis_jetzt = 0.0
36         if type(bis_jetzt) == types.FloatType:

```

```

37         neu = (bis_jetzt, 0, time.time()) # ..., Ebene 0, aktuelle Zeit
38     else:
39         bj, level, begin = bis_jetzt
40         level = level + 1
41         neu = (bj, level, begin)
42     self.zeit_in[name] = neu
43
44     def p_end(self, name):
45         bj, level, begin = self.zeit_in[name]
46         if level:
47             level = level - 1
48             neu = (bj, level, begin)
49         else:
50             neu = bj + (time.time() - begin) # Differenz zu bj hinzuzuehlen
51         self.zeit_in[name] = neu
52

```

M.4 dds-Paket

Das dds-Paket bildet eine eigenständig übersetzbare Einheit und befindet sich im CVS Baum im Verzeichnis align.

M.4.1 Das Makefile align/makefile

```

# Makefile zum Installieren des Aligners in bin
# KOKS - Studienprojekt
# Joachim Wagner, 04. Maerz 2001, 02. Mai 2001

INSTDIR=../bin/

#JAVAC=/usr/lib/jdk1.3/bin/javac
JAVAC=javac
JAVA=java
JAR=jar
CC=gcc
#CC=cc
#CFLAGS=-Wall -O2 -pedantic -m486 -malign-loops=2 -malign-jumps=2 -malign-functions=2 -DCPU=586
CFLAGS=-Wall -pedantic -m486 -malign-loops=2 -malign-jumps=2 -malign-functions=2 -DCPU=586

dummy:
    @echo "make all      kompiliert die Programme (cgaligner, ddsaligner, ddskommandozeile) lokal"
    @echo "make install   kopiert Programm nach $(INSTDIR)"
    @echo "make clean      räumt (derzeit nur dds) auf"
    @echo "make ddshelp    gibt Auskunft über die dds-Targets"

all: align dds

install: all
    cp align $(INSTDIR)

clean: dds-clean
    rm -f align align.o

align: align.c
    $(CC) $(CFLAGS) -o $@ -lm align.c

ddshelp:
    @echo "dds kompiliert (grafisches) Mavis und Kommandozeilenversion"
    @echo "ddsdemomatrix erzeugt Zufallsmatrix in der Datei zufall.mtr"

```

```

@echo "ddsclean entfernt .class und sonstige Dateien"
@echo "ddsjar erzeugt je eine .jar Datei für Mavis (mavis.jar) und Kommandozeilenaligner (al.jar)"
@echo "al.jar ist nicht komprimiert, da dies die Ladegeschwindigkeit erhöht "
@echo "ddsrunmavis zeigt, wie Mavis aus der jar-Datei zu starten ist"
@echo "ddsrunal demonstriert das gleiche für die Kommandozeilenversion"

dds: ddsbeobachtung ddsmatrix ddsmatrixAligner ddsmaper ddskommandozeile ddsjar

ddsmavis: de/denkselfbst/mavis/StutzenDialog.class de/denkselfbst/mavis/Mavis.class de/denkselfbst/mavis/MACV.

ddsmatrixAligner: de/denkselfbst/matrixAligner/AlignmentObserver.class de/denkselfbst/matrixAligner/BFSMatrixAl

ddsbeobachtung: de/denkselfbst/beobachtung/Beobachtbar.class de/denkselfbst/beobachtung/Beobachter.class

ddsmatrix: de/denkselfbst/matrix/Matrix.class de/denkselfbst/matrix/MatrixView.class de/denkselfbst/matrix/Writ

ddskommandozeile: de/denkselfbst/kommandozeile/Align.class de/denkselfbst/kommandozeile/HeadingMapper4Arno.clas

%.class: %.java
    $(JAVAC) -classpath lib/adt-20010208.jar:. $*.java

ddsjar:
    #$(JAR) cmvf de/denkselfbst/mavis/manifest mavis.jar lib/adt-20010208.jar de/denkselfbst/beobachtung/*.class
    #$(JAR) cmvf0 de/denkselfbst/kommandozeile/manifest al.jar -C lib/adt-20010208.jar de/denkselfbst/beobachtung
    $(JAR) cmf de/denkselfbst/mavis/manifest mavis.jar com/bluemarsh/adt/ de/denkselfbst/beobachtung/ de/denkself
    $(JAR) cmf0 de/denkselfbst/kommandozeile/manifest al.jar com/bluemarsh/adt/ de/denkselfbst/beobachtung/*.cla
    $(JAR) cmf0 de/denkselfbst/kommandozeile/manifest2 headingMapper.jar de/denkselfbst/beobachtung/*.class de/d

ddsclean:
    -rm -f de/denkselfbst/mavis/*.class
    -rm -f de/denkselfbst/matrix/*.class
    -rm -f de/denkselfbst/matrixAligner/*.class
    -rm -f de/denkselfbst/beobachtung/*.class
    -rm -f de/denkselfbst/kommandozeile/*.class
    -rm -f zufall.mtr
    -rm -f al.jar
    -rm -f mavis.jar
    -rm -f headingMapper.jar

ddsdemomatrix:
    @echo Schreibe 17x14 Zufallsmatrix: zufall.mtr
    $(JAVA) -classpath lib/adt-20010208.jar:. de.denkselfbst.matrix.WriteRandomMatrix 17 14 > zufall.mtr

ddsrunmavis:
    $(JAVA) -jar mavis.jar

ddsrunal:
    $(JAVA) -jar al.jar

```

M.5 Java

M.5.1 de.denkselfbst.matrix.Matrix.java

```

package de.denkselfbst.matrix;

import de.denkselfbst.beobachtung.Beobachtbar;
import de.denkselfbst.beobachtung.Beobachter;

import java.util.ArrayList;

```

```
import java.util.Arrays;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.FileReader;
```

```
/**
```

Diese Klasse repräsentiert eine aus double Werten bestehende Abstandsmatrix. Die Abstandsmatrix als solche ist die Schnittstelle zwischen Abstandsmaß und Aligner.

Daher ist diese Klasse ein sehr wichtiger und zentraler Bestandteil des Aligners. Neben den Abstandswerten enthalten Objekte dieser Klasse zwei Strings, die den Pfad zur Ausgangsdatei und Zusatzinformationen enthalten (quelle, info).

Im Allgemeinen werden Objekte dieser Klasse über einen Konstruktor mit Dateinamen aus einer Datei, bzw. aus STDIN mit Werten gefüllt. Das benutzte Dateiformat hat folgende Form:

```
Quelle-String \n
Info-String \n
Breite Hoehe \n
Wert1 \n Wert2 \n ... WertN
```

Diese Klasse erkennt neben \n auch anderen Whitespace als Trenner zwischen den Abstandswerten an. (Siehe entsprechende Methode.)

Die Abstandswerte, so wurde es im Projekt vereinbart, liegen im Intervall von 0 bis 1. Damit die Heuristik des A*-Aligners ihre Funktionalität erfüllen kann, müssen alle Abstandswerte > 0 sein. Die Methode `normalisiereDich()` verschiebt die Abstandswerte einer Matrix in das Intervall [0.1 ... 1]. Dies erhöht auch den Kontrast in einer Matrix aus TG-Werten, die in einem sehr kleinen Intervall liegen.

Eine Reihe von Methoden mit zugegebenermaßen recht ausschweifigen Namen erlauben den Zugriff auf die Verkapselten Daten.

Am wichtigsten sind `holeDatumVonStelle(int x, int y)` und `wasSindVonHierDieUnterschaetztenKostenZurRechtenUnterenEcke()`.

In dem GUI-Tool Mavis gibt es viele Objekte, die sich für Änderungen (Transformation der Daten, neue Datei geladen) interessieren, daher wurde ein einfaches Observer-Modell geschaffen (package `de.denkselbst.beobachtung`), das vom Matrix-Objekt unterstützt wird.

Sonstiges:

Die Zugriffsmethoden sind derzeit nicht synchronisiert.

```
*/
```

```
public class Matrix implements Beobachtbar{

    /** beinhaltet aus Datei gelesene Quellenangabe */
    protected String quelle;

    /** beinhaltet aus Datei gelesene Info-Zeile */
    protected String info;

    /** merkt sich den Dateinamen, dieser wird hier von der
        Methode ReadFromFile hinterlegt, bzw. von dateilos
```

```
    Konstruktoren auf "" gesetzt*/
protected String filename;

/** beinhaltet die Daten der Matrix */
protected double[][] daten;

/** merkt sich den kleinsten in der Matrix eingetragenen Wert */
protected double minimum;
/** Koordinaten des kleinsten Werts */
protected int minXPos, minYPos;

/** merkt sich den größten in der Matrix eingetragenen Wert */
protected double maximum;
/** Koordinaten des größten Werts */
protected int maxXPos, maxYPos;

/** gibt es schon einen kleinsten Wert? */
protected boolean minimumBereitsErmittelt;

/** Maximum schon ermittelt? */
protected boolean maximumBereitsErmittelt;

/**
 * Parameterloser Konstruktor:
 * sollte nur verwendet werden, wenn Matrix aus stdin gefüllt wird!
 */
public Matrix(){
    daten = new double[1][1];
    minimumBereitsErmittelt = false;
    filename="uninitialisierte Matrix!!!";
}

/** legt ein Matrix-Objekt mit einer leeren Matrix an
 * @param xDim die gewünschte Breite
 * @param yDim die gewünschte Höhe
 */
public Matrix(int xDim, int yDim){
    daten = new double[xDim][yDim];
    minimumBereitsErmittelt = false;
    filename="";
}

/**
 * legt ein Matrix-Objekt an
 * @param m die Daten
 */
public Matrix(double[][] m){
    daten = m;
    minimumBereitsErmittelt = false;
    filename="";
}

/**
 * legt ein Matrix-Objekt aus einer Datei an
 * @param datei der Dateiname
 */
public Matrix(String file){
    daten = readFromFile(file);
    minimumBereitsErmittelt = false;
```



```

}

/**
 * liest Daten aus einer Datei ein und liefert double[][]
 * (bemüht readFromBufferedReader())
 * @param file Dateiname als String
 */
private double[][] readFromFile(String file){
    filename = file; // Dateinamen merken

    try{ // BufferedReader anlegen und an zuständige Read-Methode übergeben,
        // deren Ergebnis liefern
        BufferedReader in = new BufferedReader(new FileReader(file));
        return readFromBufferedReader(in);
    }catch(IOException e){ // IO-Fehler
        System.err.println(e.getMessage());
        System.err.println("readFromFile: Fehler beim Lesen der Datei "+file+
            " . Liefere leere 6x6 Matrix.");
        return new double[6][6]; // macht readfromBufferedReader doch eh?!
    }
}

}

/**
 * liest Daten von StdIn ein und liefert double[][]
 * (bemüht readFromBufferedReader())
 */
private double[][] readFromStdIn(){
    filename="von_stdin_gelesen";
    BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
    return readFromBufferedReader(in);
}

}

/**
 * liest Daten von einem BufferedReader ein und liefert double[][]
 * (bemüht readFromBufferedReader())
 */
private double[][] readFromBufferedReaderJoachim(BufferedReader bf){
    filename="von_BufferedReader_gelesen";
    return readFromBufferedReader(bf);
}

}

/**
 * liest Daten aus BufferedReader und liefert double[][]
 * (macht die eigentliche Arbeit für die beiden anderen Read-Methoden)
 * @param in der BufferedReader
 */
private double[][] readFromBufferedReader(BufferedReader in){
    int x = 0; // Abmessungen der Matrix
    int y = 0; // "
    double[][] matrix = new double[0][0]; // Referenz auf Matrix

    try{
        quelle = in.readLine();
        info = in.readLine();
        String line = in.readLine();
        StringTokenizer tok = new StringTokenizer(line, "\t \n", false);

```

```

// nächstes Token besorgen
if(!tok.hasMoreTokens()){
    while(!tok.hasMoreTokens() && line!=null){
        line = in.readLine();
        tok = new StringTokenizer(line, "\t \n", false);
    }
}
// breite der Matrix holen
x = Integer.parseInt(tok.nextToken());

// nächstes Token besorgen
if(!tok.hasMoreTokens()){
    while(!tok.hasMoreTokens() && line!=null){
        line = in.readLine();
        tok = new StringTokenizer(line, "\t \n", false);
    }
}
// Höhe der Matrix holen
y = Integer.parseInt(tok.nextToken());

// Matrix anlegen
matrix = new double[x][y];
String token = null;

// Zeilenweise einlesen
for (int j = 0; j < y; j++){
    for (int i = 0 ; i < x ; i++){
        // Nächstes Token besorgen
        if(!tok.hasMoreTokens()){
            while(!tok.hasMoreTokens() && line!=null){
                line = in.readLine();
                tok = new StringTokenizer(line, "\t \n", false);
            }
        }
        token = tok.nextToken();
        matrix[i][j] = Double.parseDouble(token);
        //System.err.println("habe "+(i+(j*x))+ " von "+(x*y)+
        // " Zahlen eingelesen.");
    }
}
} catch(IOException e){
    System.err.println(e.getMessage());
    System.err.println(" ReadFromBufferedReader: IO-Fehler. Liefere leere"+
        " 6x6 Matrix.");
    return new double[6][6];
}

return matrix; // Die Matrix liefern
}

/**
 * liefert den Quellen-String
 */
public String holeQuelle(){
    return quelle;
}

```

```
/**
 * liefert den Info-String
 */
public String holeInfo(){
    return info;
}

/**
 * liefert den Dateinamen als String
 */
public String holeDateiname(){
    return filename;
}

/**
 * liefert die Breite der Matrix
 */
public int holeBreite(){
    return daten.length;
}

/**
 * liefert die Hoehe der Matrix
 */
public int holeHoehe(){
    return daten[0].length;
}

/**
 * Kopie von daten[][] liefern. Eine solche Kopie der daten[][] kann von
 * anderen Objekten ohne Auswirkungen auf andere Programmteile destruktiv verändert
 * werden.
 */
public double[][] kopieDerDaten(){
    double[][] kopie = new double[holeBreite()][holeHoehe()];

    for(int i = 0; i < holeBreite(); i++){
        for(int j = 0; j < holeHoehe(); j++){
            kopie[i][j] = daten[i][j];
        }
    }

    return kopie;
}

/**
 * liefert den Wert der Matrix an der Stelle (x,y)
 * @param x die x-Position
 * @param y die y-Position
 */
public double holeDatumVonStelle(int x, int y){
    return daten[x][y];
}

/**
 * setzt den Wert der Matrix an der Stelle (x,y) auf datum
 * Minimum und Maximum werden gepflegt
```

```

    * @param x die x-Position
    * @param y die y-Position
    * @param datum der zu setzende double Wert
    */
public void setzeDatumAnStelle(int x, int y, double datum){
    // datum eintragen
    daten[x][y] = datum;
    // handelt es sich um ein neues Minimum?
    if (minimumBereitsErmittelt && datum < minimum) minimum = datum;
    if (maximumBereitsErmittelt && datum > maximum) maximum = datum;
    // Beobachtern Meldung von der Änderung der Matrix machen...
    beobachterBenachrichtigen("Habe "+datum+
        " an Stelle ("+x+", "+y+") gesetzt.");
}

/**
 * intern aufrufen, um Zustandsänderung klarzumachen
 * altes Minimum und Maximum verlieren Gültigkeit
 * z.B. nach Transformation und Stutzen!
 */
protected void geaendert(){
    minimumBereitsErmittelt = false;
    maximumBereitsErmittelt = false;
    //minimum = -999; // Weg damit!
    //maximum = +999; // "
    //minXPos = -666;
    //minYPos = -666;
}

/**
 * Kleinsten Wert (Minimum) der Matrix liefern
 */
public double wasIstDerKleinsteWert(){
    if(!minimumBereitsErmittelt){
        minimum = besorgeMinimum();
        minimumBereitsErmittelt = true;
    }
    return minimum;
}

/** x-Koordinate des Minimums liefern */
public int getMinX(){
    besorgeMinimum(); // Falls Wert nicht im Cache, Besorgung veranlassen.
    return minXPos;
}

/** y-Koordinate des Minimums liefern */
public int getMinY(){
    besorgeMinimum(); // Falls Wert nicht im Cache, Besorgung veranlassen.
    return minYPos;
}

/**
 * Matrix zellenweise durchgehen und dabei den
 * kleinsten Wert (Minimum) bestimmen
 */
private double besorgeMinimum(){
    minXPos = 0;
    minYPos = 0;
}

```

```

double min = daten[0][0];
for(int i = 0; i < holeBreite(); i++){
    for(int j = 0; j < holeHoehe(); j++){
        if(daten[i][j] < min){
            min = daten[i][j];
            minXPos = i;           // Koordinaten merken
            minYPos = j;           // "
        }
    }
}
return min;
}

/**
 * Maximum der Matrix liefern
 *
 */
public double wasIstDerGroessteWert(){
    if(!maximumBereitsErmittelt){
        maximum = besorgeMaximum();
        maximumBereitsErmittelt = true;
    }
    return maximum;
}

/**
 * Matrix zellenweise durchgehen und dabei den
 * größten Wert (Maximum) aufsammeln
 */
private double besorgeMaximum(){
    double max = daten[0][0];
    for(int i = 0; i < holeBreite(); i++){
        for(int j = 0; j < holeHoehe(); j++){
            if(daten[i][j] > max){
                max = daten[i][j];
                maxXPos = i;       // Koordinaten merken
                maxYPos = j;       // "
            }
        }
    }
    return max;
}

/**
 * Unterschätzte Kosten zur rechten unteren Ecke liefern
 * (Diagonale Distanz)
 */
public double wasSindVonHierDieUnterschaetztenKostenZurRechtenUnterenEcke(
    int xPos, int yPos){
    return Math.max( Math.abs((holeBreite()-1) - xPos),
        Math.abs((holeHoehe()-1) - yPos) )
        * wasIstDerKleinsteWert();
}

/**
 * Mindestkosten von x1,y1 nach x2,y2 liefern
 */

```

```

public double minKost(int x1, int y1, int x2, int y2){
    return Math.max( Math.abs((x2-x1)),
        Math.abs(y2-y1)) *
        wasIstDerKleinsteWert();
}

/**
 * minimale Anzahl Schritte von xPos, yPos zum Ziel
 * (rechte untere Ecke)
 */
public int wievieleSchritteVonHierZurRechtenUnterenEcke(int xPos, int yPos){
    return Math.max( Math.abs((holeBreite()-1) - xPos),
        Math.abs((holeHoehe()-1) - yPos) );
}

/**
 * Matrix auf die gegebenen Dimensionen zurechtstutzen
 */
public void stutzenAuf(int x, int y){
    // bei Gelegenheit Sicherheitsüberprüfungen einbauen!

    double[][] neu = new double[x][y];
    for(int i = 0; i < x; i++){
        for(int j = 0; j < y; j++){
            neu[i][j]=daten[i][j];
        }
    }

    daten = neu;
    geaendert(); // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
    // die Beobachter benachrichtigen
    beobachterBenachrichtigen("Habe die Matrix auf "+x+" * "+y+" gestutzt.");
}

/**
 * Eine komplette Spalte der Matrix auf Wert w setzen
 */
public void fillColumn(int x, double w){
    if(0 <= x && x < holeBreite()){
        for(int j = 0; j < holeHoehe(); j++){
            daten[x][j] = w;
        }
        geaendert(); // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
        // die Beobachter benachrichtigen
        beobachterBenachrichtigen("Habe alle Zellen der Spalte"+x+" auf "+w+
            " gesetzt.");
    }
}

/**
 * Eine komplette Zeile der Matrix auf Wert w setzen
 */
public void fillRow(int y, double w){
    if(0 <= y && y < holeHoehe()){
        for(int i = 0; i < holeBreite(); i++){
            daten[i][y] = w;
        }
        geaendert(); // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
        // die Beobachter benachrichtigen
        beobachterBenachrichtigen("Habe alle Zellen der Zeile"+y+" auf "+w+
            " gesetzt.");
    }
}

```

```

    }
}

/**
 * Mittelwert berechnen
 */
public double berechneMittelwert(){
    double sum = 0;
    for(int i = 0; i < holeBreite(); i++){
        for(int j = 0; j < holeHoehe(); j++){
            sum+=daten[i][j];
        }
    }
    return sum / (holeBreite() * holeHoehe());
}

/**
 * Varianz berechnen
 */
public double berechneVarianz(){
    double mittelwert = berechneMittelwert();

    double sum = 0;
    for(int i = 0; i < holeBreite(); i++){
        for(int j = 0; j < holeHoehe(); j++){
            sum+= ((daten[i][j] - mittelwert) * (daten[i][j] - mittelwert));
        }
    }
    return sum / ((holeBreite() * holeHoehe())-1);
}

/**
 * Standardabweichung berechnen
 */
public double berechneStandardabweichung(){
    return Math.sqrt(berechneVarianz());
}

/**
 * Transformation: Daten normalisieren auf Intervall [0.1 .. 1]
 */
public void normalisiereDich(){
    int breite = holeBreite();
    int hoehe = holeHoehe();
    double min = wasIstDerKleinsteWert();
    double max = wasIstDerGroessteWert();

    double transformiert[][] = new double[breite][hoehe];

    for(int i = 0; i < holeBreite(); i++){
        for(int j = 0; j < holeHoehe(); j++){
            // in Bereich [0 .. 1] skalieren
            if(max-min != 0){
                transformiert[i][j] = (daten[i][j] - min) * (1.0 / (max - min));
            }else transformiert[i][j]=42;
        }
    }
}

```

```

        // in Bereich [0.1 .. 1] verschieben
        transformiert[i][j] = transformiert[i][j] * (0.9) + 0.1;
    }
}
daten = transformiert;
geaendert(); // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
beobachterBenachrichtigen("Matrixwerte auf Intervall [0.1 .. 1] skaliert");
}

/**
 * Daten transformieren: Clusterbildung nach Rangfolge der Daten
 */
public void rangfolgenClustering(int anzahlKlassen){

    int breite = holeBreite();
    int hoehe = holeHoehe();
    int anzahlWerte = breite * hoehe;

    // jede Klasse bekommt als Wert ein Vielfaches von diesem Wert
    double werteSchrittweite = 1.0 / (anzahlKlassen + 1); // +1 für eine
        // Dummy-Klasse, damit kein Wert mehr auf 0.0 abgebildet wird...

    double[] werte = new double[anzahlWerte];
    double[][] neu = new double[breite][hoehe];

    for(int i = 0; i < breite; i++){
        for(int j = 0; j < hoehe; j++){
            werte[(j*breite)+i] = daten[i][j];
        }
    }

    // So, Werte sind sortiert!
    Arrays.sort(werte);
    // Alle Werte durchgehen
    int klasse = 1;
    int kCount = 0;
    int werteProKlasse = anzahlWerte / anzahlKlassen;
    for(int i = 0; i < werte.length; i++){
        kCount++;
        if(kCount > werteProKlasse){ klasse++; kCount=0;}

        repl(werte[i],klasse * werteSchrittweite, neu);
    }
    daten = neu;
    geaendert(); // Matrix mitteilen, das Min, Max weggeworfen werden müssen...
    beobachterBenachrichtigen("bogoRangfolgenClustering durchgeführt");
}

/** schlampige replace-Funktion; ersetzt alle Vorkommen von w1
 * in der Matrix durch w2 im Argument
 */
private void repl(double w1, double w2, double[][] x){
    for(int i = 0; i < holeBreite(); i++){
        for(int j = 0; j < holeHoehe(); j++){
            if(daten[i][j]==w1) x[i][j]=w2;
        }
    }
}
}

```



```

/**
 * Matrix mit neuen Daten aus Datei füttern
 * @param file String mit Dateiname
 */
public void setzeDatenAusDatei(String file){
    // Daten besorgen
    daten = readFromFile(file);
    // Kleinster Wert dieser neuen Daten ist logischerweise noch nicht ermittelt
    // worden
    minimumBereitsErmittelt = false;
    // Beobachtern Meldung davon machen...
    beobachterBenachrichtigen("Habe potentiell neue Daten aus Datei "+file+
        " eingelesen.");
}

/**
 * Matrix mit neuen Daten aus Stdin füttern
 */
public void setzeDatenAusStdin(){
    BufferedReader bf = new BufferedReader(new
        InputStreamReader(System.in));
    setzeDatenAusGepuffertemLeser(bf);
}

/**
 * Matrix mit neuen Daten aus BufferedReader füttern
 */
public void setzeDatenAusGepuffertemLeser(BufferedReader br){
    // Daten besorgen
    daten = readFromBufferedReaderJoachim(br);
    // Kleinster Wert dieser neuen Daten ist logischerweise noch nicht ermittelt
    // worden
    minimumBereitsErmittelt = false;
    // Beobachtern Meldung davon machen...
    beobachterBenachrichtigen("Habe potentiell neue Daten von stdin gelesen.");
}

/**
 * Matrix als String liefern (billige Ausgabe zu Diagnosezwecken)
 */
public String toString(){
    int b = holeBreite();
    int h = holeHoehe();
    StringBuffer buf = new StringBuffer("Matrix \n   Breite: "+b+
        "   Höhe: "+h+"\n   Daten\n");

    for(int j = 0; j < h; j++){
        buf.append("   ");
        for(int i = 0; i < b; i++){
            buf.append( holeDatumVonStelle(i,j) +" \t");
        }
        buf.append("\n");
    }

    return buf.toString();
}

/**

```

```

    * nimmt die Objekte auf, die Interesse an
    * Änderungen der Matrix angemeldet haben
    */
private ArrayList dieBeobachter;

/**
 * Einen neuen Beobachter in die Liste der im Falle einer
 * Änderung zu benachrichtigen Objekte aufnehmen.
 * @param b das Objekt, das Interesse an den Änderungen der Matrix hat
 */
public void beobachterAnmelden(Beobachter b){
    if(dieBeobachter == null) dieBeobachter = new ArrayList();
    dieBeobachter.add(b);
}

/**
 * einen Beobachter aus der Liste der zu benachrichtigen
 * Objekte entfernen.
 * @param b das Objekt, das das Interesse an den
 * Änderungen der Matrix verloren hat
 */
public void beobachterAbmelden (Beobachter b){
    if(dieBeobachter != null) dieBeobachter.remove(b);
}

/**
 * alle Beobachter ob einer Änderung benachrichtigen, Referenz
 * auf das Beobachtete Objekt (this)
 * und Nachricht String werden mitgeliefert
 * @param nachricht String, der zur Beschreibung der Änderung
 * benutzt werden kann
 */
public void beobachterBenachrichtigen(String nachricht){

    if(dieBeobachter == null) return; // keine Beobachter, keine Nachricht.

    for(int i=0; i<dieBeobachter.size(); i++){
        if(dieBeobachter.get(i) != null){ // Wenn da überhaupt jemand ist (sollte
            //immer so sein!)
                ((Beobachter)dieBeobachter.get(i)).beobachtetesObjektHatSichGeregt(this,
nachricht);
            }
        else{ // NULL-Referenzen löschen! sollte alles nie passieren!!!
            System.out.println("Matrix: NULL-Beobachter entfernt!");
            dieBeobachter.remove(i);
        }
    }
}

/** winziger Test: Lese Matrix aus Datei args[0] ein
 * und gebe sie auf System.out aus.
 */
public static void main (String[] args){
    Matrix m = new Matrix(args[0]);
    System.out.println(m);
}
}

```

M.5.2 de.denkselbst.matrix.Matrix.Pfad.java

```

package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.Iterator;

/**
Diese Klasse repräsentiert eine Liste von Koordinatenpaaren.

Ein Pfad ist eine Liste von Koordinatenpaaren. Im Rahmen von
Graphensuchalgorithmen ist das ein geeignetes Mittel, um einen Pfad in einer
Matrix zu beschreiben. Längere horizontale Abschnitte in einem Pfad werden als
n:m - Alignments aufgefasst, durch eine Konvertierung des Pfades in eine
KoksZuordnung wird dem Rechnung getragen.
MatrixAligner liefern ihre Ergebnisse derzeit als Pfade.

Pfade wissen, wie sie sich in KoksZuordnungen konvertieren können.

Die Klasse KoksZuordnung bietet eine Konstruktor, der einen Pfad
entgegen nimmt. Die KoksZuordnung ruft dann die Methode toWagnerString beim
Pfad auf und erhält so einen String, den sie interpretieren kann. Quelle und
Info müssen aus einer Matrix übernommen werden, deshalb hat die
toWagnerString Methode Argumente, in denen diese Strings übergeben werden
müssen.

*/

public class Pfad{

    /**
     * Kapselt x- und y-Koordinate
     */
    public static class Koordinate{
        /** enthält x-Koordinate*/
        public int x;
        /** enthält y-Koordinate*/
        public int y;

        /**
         * legt eine Koordinate an
         * @param x - x-Koordinate
         * @param y - y-Koordinate
         */
        public Koordinate(int x, int y){
            this.x = x;
            this.y = y;
        }
    }

    /** beinhaltet die Koordinaten, aus denen der Pfad besteht */
    ArrayList p;

    /** neues Pfad Objekt anlegen */
    public Pfad(){
        p = new ArrayList();
    }

    /** eine Koordinate ans Ende des Pfades anhängen

```

```

    * @param x - x-Koordinate
    * @param y - y-Koordinate
    */
public void haengeAn(int x, int y){
    p.add(new Koordinate(x, y));
}

/** eine Koordinate an den Anfang des Pfades anhängen
 * @param x - x-Koordinate
 * @param y - y-Koordinate
 */
public void haengeVorneAn(int x, int y){
    p.add(0, new Koordinate(x, y));
}

/** einen Iterator über die Elemente des Pfades liefern*/
public Iterator iterator(){
    return p.iterator();
}

/** Pfade umdrehen*/
public void umdrehen(){
    if(p.size() == 0) return;
    ArrayList p2 = new ArrayList(p.size());
    for(int i = 0; i < p.size(); i++){
        p2.add( p.get(p.size()-1-i));
    }
    p = p2;
}

/**
 * Einfache Ausgabe
 */
public String toString(){
    StringBuffer buf = new StringBuffer();
    Koordinate k;
    for(int i = 0; i < p.size(); i++){
        k = (Koordinate)p.get(i);
        buf.append("[ "+k.x+", "+k.y+" ] ");
    }
    return buf.toString();
}

/**
 * Pfad als String im Wagner-Format liefern.
 * Verwendungszweck ist sowohl textuelle Ausgabe, als auch
 * Schritt bei der Konvertierung in KoksZuordnung.
 * Die Strings quelle und info müssen übergeben werden, Matrix-
 * Objekte verfügen über entsprechende Strings, von dort sollten
 * diese Informationen also beschafft werden.
 *
 * Der Algorithmus verfolgt den Pfad von Anfang bis Ende und verwandelt
 * ihn in eine Folge von Segment-Zuordnungen.
 * Eine Diagonalbewegung nach rechts unten schliesst die aktuelle
 * Segmentzuordnung ab und beginnt eine neue. Durch vertikalen oder
 * horizontalen Pfadverlauf verknüpfte Sätze werden an die aktuelle
 * Segment-Zuordnung angehängt.
 */
public String toWagnerString(String quelle, String info){
    if(p.size()==0) return ""; // leerer Pfad = leerer Wagner-String

```

```

StringBuffer buf = new StringBuffer(quelle+"\n"+info+"\n");

int segment = 0;
Koordinate k;

int RECHTS = 1;           // einige Konstanten
int RUNTER = 2;
int DIAGONAL = 3;
int UNBEKANNT = -9;

// Vorbereitungen
ArrayList seg1 = new ArrayList(2); // Satznummern Sprache 1
ArrayList seg2 = new ArrayList(2); // Satznummern Sprache 2

k = (Koordinate)p.get(0);
int altX = k.x;
int altY = k.y;

seg2.add(new Integer(k.x));
seg1.add(new Integer(k.y));

int altRichtung = UNBEKANNT;
int neueRichtung = UNBEKANNT;

// alle Koordinaten abwandern
for(int i = 1; i < p.size(); i++){

    k = (Koordinate)p.get(i);

    int neuesX = k.x;
    int neuesY = k.y;

    // Laufrichtung ermitteln
    if( (neuesX == altX) && (neuesY != altY) ){ // Der Pfad führt nach unten
        neueRichtung = RUNTER;
    }
    else if( (neuesX != altX) && (neuesY == altY) ){// nach rechts
        neueRichtung = RECHTS;
    }
    else{ // Diagonal
        neueRichtung = DIAGONAL;
    }

    // Wenn's diagonal geht, dann sofort neue Ausgabe!
    if(neueRichtung == DIAGONAL){
        // Ausgabe, Listen zurücksetzen
        // Anfang auseben
        buf.append(""+segment+"\t[");
        for(int t=0; t<seg1.size(); t++){
            buf.append(""+seg1.get(t)+",");
        }
        // Überschüssiges Komma entfernen
        buf.deleteCharAt(buf.length()-1);
        // Nächste Spalte
        buf.append("]\t[");
        for(int t=0; t<seg2.size(); t++){
            buf.append(""+seg2.get(t)+",");
        }
        // Überschüssiges Komma entfernen
        buf.deleteCharAt(buf.length()-1);
        // Klammer zu und \newline

```

```

        buf.append("]\n");

        // Nächste Runde nächstes Segment verarzten
        segment++;

        seg1 = new ArrayList(2); // Satznummern Sprache 1
        seg2 = new ArrayList(2); // Satznummern Sprache 2

        altRichtung = UNBEKANNT;

        seg2.add(new Integer(k.x));
        seg1.add(new Integer(k.y));
    }
    else if(altRichtung == UNBEKANNT){ // alte Richtung unbekannt
        if(neueRichtung == RUNTER){
            seg1.add(new Integer(k.y));
        }

        if(neueRichtung == RECHTS){
            seg2.add(new Integer(k.x));
        }

        altRichtung = neueRichtung;
    }
    else{ // Alte Richtung bekannt

        if( neueRichtung == RECHTS){
            seg2.add(new Integer(k.x));
        }
        else if(neueRichtung == RUNTER){
            seg1.add(new Integer(k.y));
        }
    }
}

altX = k.x; // Koordinaten für die nächste Iteration aufheben
altY = k.y; // "
} // Schleifenende

// Ausgeben, was noch in den Puffern steht.
// Anfang ausgeben
buf.append(""+segment+"\t[");
for(int t=0; t<seg1.size(); t++){
    buf.append(""+seg1.get(t)+",");
}
// Überschüssiges Komma entfernen
buf.deleteCharAt(buf.length()-1);
// Nächste Spalte
buf.append("]\t[");
for(int t=0; t<seg2.size(); t++){
    buf.append(""+seg2.get(t)+",");
}
// Überschüssiges Komma entfernen
buf.deleteCharAt(buf.length()-1);
// Klammer zu und \newline
buf.append("]\n");

```

```
    return buf.toString(); // Ergebnis liefern
}

/**
 * Demonstration / Test
 */
public static void main(String[] args){
    System.out.println("Test 1");
    Pfad p = new Pfad();
    p.haengeAn(0,0);
    p.haengeAn(0,1);
    p.haengeAn(1,2);
    p.haengeAn(2,3);
    p.haengeAn(2,4);
    p.haengeAn(3,5);
    p.haengeAn(3,6);

    System.out.println("toString:");
    System.out.println(p.toString());

    System.out.println("");
    System.out.println("toWagner:");
    System.out.println(p.toWagnerString("generiert","Test"));

    p = new Pfad();
    p.haengeAn(0,0);
    p.haengeAn(1,0);
    p.haengeAn(2,1);
    p.haengeAn(3,1);
    p.haengeAn(4,1);
    p.haengeAn(5,2);
    p.haengeAn(6,2);

    System.out.println("");
    System.out.println("Test 2");
    System.out.println("toString:");
    System.out.println(p.toString());

    System.out.println("");
    System.out.println("toWagner:");
    System.out.println(p.toWagnerString("generiert","Test"));

    p = new Pfad();
    p.haengeAn(0,0);
    p.haengeAn(1,1);
    p.haengeAn(2,2);
    p.haengeAn(3,3);
    p.haengeAn(4,4);
    p.haengeAn(5,5);
    p.haengeAn(6,6);

    System.out.println("");
    System.out.println("Test 3");
    System.out.println("toString:");
    System.out.println(p.toString());

    System.out.println("");
```

```

        System.out.println("toWagner:");
        System.out.println(p.toWagnerString("generiert", "Test"));
    }
}

```

M.5.3 de.denkselbst.matrix.Matrix.KoksZuordnung.java

```

package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.StringTokenizer;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.StringReader;

/**
Diese Klasse repräsentiert ein Alignment. Ein Alignment besteht aus
Zuordnungen (KoksZuordnung.Zuordnung), die n Sätze (Segmente) aus L1 mit m Sätzen aus L2
verbinden. Eine Liste solcher Zuordnungen, die sequentiell die Menge aller
Sätze bedeckt, ist eine KoksZuordnung.
KoksZuordnungen können im Wagner-Format textuell ausgegeben werden und
stellen damit die Schnittstelle vom Aligner zu den nachfolgenden
Modulen dar.
*/

public class KoksZuordnung {

    /* Innere Klasse verkapselt die Zuordnung
    * von zwei Satzmengen.
    */
    public static class Zuordnung{

        private ArrayList lang1, lang2;

        /** Konstruktor */
        public Zuordnung(){
            lang1 = new ArrayList(2);
            lang2 = new ArrayList(2);
        }

        /** Füge Liste 1 eine (Satz)Nummer hinzu*/
        public void addL1(int i){
            lang1.add(new Integer(i));
        }

        /** Füge Liste 2 eine (Satz)Nummer hinzu*/
        public void addL2(int i){
            lang2.add(new Integer(i));
        }

        /** Liefere Listel nach aussen (z.B. wird damit an anderer Stelle
        eine HTML-Tabelle mit den tatsächlichen Sätzen erzeugt) */
        public ArrayList getL1(){
            return lang1;
        }

        /** Liefere Liste2 nach aussen (z.B. wird damit an anderer Stelle

```



```

    eine HTML-Tabelle mit den tatsächlichen Sätzen erzeugt) */
public ArrayList getL2(){
    return lang2;
}

/** Gibt die Zuordnung im Wagnerformat aus*/
public String toString(){
    // Auch hier wird erst die Y-Seite, dann die X-Seite ausgegeben,
    // damit das Ergebnis wieder stimmt...
    // Nicht geprüft, ob die Vertauschung hier wirklich notwendig,
    // wenn es also wieder Probleme gibt, erst hier nachsehen!

    StringBuffer buf = new StringBuffer("[");

    for(int i = 0; i < lang2.size(); i++){
        buf.append( lang2.get(i) );
        buf.append(",");
    }
    buf.deleteCharAt(buf.length()-1);

    buf.append("]\t[");

    for(int i = 0; i < lang1.size(); i++){
        buf.append( lang1.get(i) );
        buf.append(",");
    }
    buf.deleteCharAt(buf.length()-1);

    buf.append("]");

    return buf.toString();
}

/**
 * Trägt in der Liste koord die entsprechenden Koordinaten der Zuordnung ein
 * x1,y1,x2,y2,...xn,yn
 */
public void betroffeneKoordinaten(ArrayList koord){

    for(int i = 0; i < lang1.size(); i++){
        for(int j = 0; j < lang2.size(); j++){
            // Y ist X und X ist Y, habe ich gerade bemerkt...
            // Also erst kommen in der eingelesenen Datei Y-Werte, dann X-Werte
            // Da ich die falschrüm einlese, gebe ich sie hier auch falschrüm wieder
            // aus, und dann ist ALLES richtig... :-)
            koord.add(lang2.get(j));
            koord.add(lang1.get(i));
        }
    }
}

}

private String quelle;
private String info;

private ArrayList zuordnungen;

/**
 * Aus Datei Koksuzuordnung einlesen

```

```
    */
    public KoksZuordnung(String file){
        zuordnungen = new ArrayList(30);
        readFromFile(file);
    }

    /**
     * Konstruktor: aus Pfad p KoksZuordnung erzeugen
     * @param p Pfad
     * @param quelle = matrix.holeQuelle();
     * @param info = matrix.holeInfo();
     */
    public KoksZuordnung(Pfad p, String quelle, String info){
        zuordnungen = new ArrayList(30);
        String s = p.toWagnerString(quelle, info);
        readFromString(s);
    }

    /** Den ganzen Info kram mal in ein Interface/Klasse verpacken, das nutzen viele Klassen!*/
    /** Info-String liefern */
    public String holeInfo(){
        return info;
    }

    /** Quelle liefern*/
    public String holeQuelle(){
        return quelle;
    }

    /**
     * liest Daten aus einer Datei im Wagner-Format ein und liefert ein entsprechendes Objekt
     * @param file Dateiname als String
     */
    private void readFromFile(String file){
        try{
            BufferedReader in = new BufferedReader(new FileReader(file));
            readFromBufferedReader(in);
        }catch(FileNotFoundException e){
            System.err.println("IO-Fehler. (file not found)");
        }
    }

    /**
     * liest Daten im WagnerFormat aus einem String ein
     */
    private void readFromString(String s){
        BufferedReader in = new BufferedReader(new StringReader(s) );
        readFromBufferedReader(in);
    }

    /**
     * Hier wird die eigentliche Einlesarbeit gemacht...
     */
    private void readFromBufferedReader(BufferedReader in){
        int segment;
        String lang1;
```

```

String lang2;

StringBuffer l1,l2;

Character komma = new Character(',');

try{

    quelle = in.readLine();        // Quelle und
    info = in.readLine();        // Info lesen

    String line = in.readLine(); //

while (line != null){
    StringTokenizer tok = new StringTokenizer(line, "\t \n", false);

    // Segmentnummer holen
    segment = Integer.parseInt(tok.nextToken());

    // neue Zuordnung anlegen
    Zuordnung z = new Zuordnung();

    // Liste der Sätze der ersten Sprache holen
    l1 = new StringBuffer();
    l1.append(tok.nextToken());
    // Solange am Ende des Puffers ein "," rumhängt
    while(new Character(l1.charAt(l1.length()-1)).toString().equals(",")){
        l1.append(tok.nextToken());
    }
    lang1 = l1.toString();        // nur für die Ausgabe...
    // l1 in die Satznummern zerlegen

    if(new Character(l1.charAt(0)).compareTo(new Character('[')) == 0){
        l1.deleteCharAt(0);

        while(new Character(l1.charAt(0)).compareTo(new Character('[')) != 0){
            StringBuffer number = new StringBuffer();
            while(Character.isDigit(l1.charAt(0))){
                number.append(l1.charAt(0));
                l1.deleteCharAt(0);
            }

            // Lang1-Satzzahl in z aufnehmen
            z.addL1(Integer.parseInt(number.toString()));

            if(new Character(l1.charAt(0)).compareTo(new Character(',')) == 0)
                l1.deleteCharAt(0);

        }
    }

    // Liste der Sätze der zweiten Sprache holen
    l2 = new StringBuffer();
    l2.append(tok.nextToken());
    // Solange am Ende des Puffers ein "," rumhängt
    while(new Character(l2.charAt(l2.length()-1)).toString().equals(",")){
        l2.append(tok.nextToken());
    }
    lang2 = l2.toString();

```

```

        if(new Character(l2.charAt(0)).compareTo(new Character('[') == 0){
l2.deleteCharAt(0);

while(new Character(l2.charAt(0)).compareTo(new Character(']')) != 0){
    StringBuffer number = new StringBuffer();
    while(Character.isDigit(l2.charAt(0))){
        number.append(l2.charAt(0));
        l2.deleteCharAt(0);
    }
    // Lang2-Satzzahl in z aufnehmen
    z.addL2(Integer.parseInt(number.toString()));

    if(new Character(l2.charAt(0)).compareTo(new Character(',')) == 0)
        l2.deleteCharAt(0);
}
}

// So, die Zuordnung dieses Segmentes ist fertig eingelesen.
zuordnungen.add(z);
line = in.readLine();
}

} catch(IOException e){
    System.err.println(e.getMessage());
    System.err.println("KoksZuordnung.readFromBufferedReader: IO-Fehler beim lesen aus BufferedReader "+
}

}

/** Erzeugt Wagner-Format (Datei, nicht Pizza) */
public String toString(){

    StringBuffer buf = new StringBuffer();

    for(int i = 0; i < zuordnungen.size(); i++){
        Zuordnung z = (Zuordnung)zuordnungen.get(i);

        buf.append(z.toString());
        buf.append("\n");
    }

    return buf.toString();
}

/**
 * Liefert die Liste der Koordinaten, die im Alignment liegen
 * x,y,x1,y1,x2,y2,...xn,yn
 */
public ArrayList betroffeneKoordinaten(){
    ArrayList koord = new ArrayList();
    for(int i = 0; i < zuordnungen.size(); i++){
        Zuordnung z = (Zuordnung)zuordnungen.get(i);

        z.betroffeneKoordinaten(koord);

```

```

    }
    return koord;
}

/**
 * Zum Test mal die betroffenen Koordinaten ausgeben
 */
public void printKoord(){
    ArrayList x = betroffeneKoordinaten();

    for(int i=0; i < x.size(); i+=2){
        System.out.println(" "+x.get(i)+" "+x.get(i+1)+"");
    }
}

/**
 * Zuordnungen rausrücken
 * (Wird benutzt um HTML-Tabelle in HTMLKocher zu bauen...)
 */
public ArrayList getZuordnungen(){
    return zuordnungen;
}

/** winziger Test: Lese Matrix aus Datei args[0] ein und gebe sie auf System.out aus. */
public static void main (String[] args){
    // Wagner-Datei einlesen und KoksZuordnung daraus kochen
    KoksZuordnung k = new KoksZuordnung(args[0]);

    // Zuordnung ausgeben
    System.out.println(k);

    // Diejenigen Koordinaten ausgeben, die von der Zuordnung betroffen sind
    k.printKoord(); // Die können dann in Mavis eingezeichnet werden...
}
}

```

M.5.4 de.denkselbst.matrix.Matrix.MatrixView.java

```

package de.denkselbst.matrix;

import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrix.Matrix;
import de.denkselbst.beobachtung.Beobachter;
import de.denkselbst.mavis.ShowHTMLFrame;

import java.io.*;
import java.net.URL;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.JViewport;

```

```

import java.awt.geom.Point2D;
import java.awt.Graphics;
import java.awt.Color;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.Image;
import java.awt.Toolkit;
import java.util.ArrayList;
import java.util.Iterator;
// für Table-Model
import javax.swing.table.AbstractTableModel;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.DefaultCellEditor;
import javax.swing.table.TableCellRenderer; import javax.swing.JLabel;
import javax.swing.JDialog;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JColorChooser;
import javax.swing.BorderFactory;
import javax.swing.border.Border; import javax.swing.JScrollPane;
import javax.swing.JFrame;
import javax.swing.SwingUtilities;
import java.awt.*;
import java.awt.event.*;
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.AbstractTableModel;
import javax.swing.table.TableColumn;
import javax.swing.ListSelectionModel;

/**
 * Diese Klasse stellt eine Matrix grafisch dar und stellt Fenster für
 * die Alignmentpfade bereit. Fast alles
 * sichtbare in Mavis stammt von hier.
 */

public class MatrixView extends JPanel implements Beobachter, MouseMotionListener, MouseListener{

    /** Referenz auf die anzuzeigende Matrix */
    private Matrix matrix;
    /** Liste von verschiedenen Alignmentpfaden(/KoksZuordnungen) als Elem-Objekte verpackt..*/
    private ArrayList pfade;

    /** Innere Klasse: Bündelt Pfad/KoksZuordnung mit Farbe, Info und Sichtbarkeitsstatus */
    public class Elem{
        Object object;
        boolean visible;
        Color color;
        String info;

        /** Konstruktor */
        public Elem(Object o, boolean vis, Color c, String i){
            object = o;
            visible = vis;
            color = c;
            info = i;
        }
    }
}

```

```

/** Referenz auf Objekt innerer Klasse T, das die Pfade einer JTable anzeigbar macht*/
T tableModel;
/** zugehörige JTable*/
JTable table;

/** Abmessungen der gezeichneten Matrixfelder */
private int breite, hoehe;
/** Hält Hintergrundbild für besseres Aussehen, wenn keine Matrix vorhanden ist (nur Spielerei...)*
private Image hgBild;

/** Flagge: Diagonale zeichnen oder nicht */
private boolean zeigeDiag = false;

/** Referenz auf Fenster, dass zu Matrixzellen Werte anzeigt*/
private XYInfo xyinfo;
/** Zeigt wenn möglich tatsächliche Sätze an (X,Y) an (wenn Korpus aus Datei stammt...)*
private SatzInfoDialog satzinfo;
/** Zeigt wenn möglich Abstandinfo an (X,Y) an */
private AbstandInfoDialog abstandinfo;
/** Dialog mit weiteren Infos zur Matrix*/
private MatrixStatInfoDialog matrixInfo;

/**
 * legt neue MatrixView ohne Inhalt an
 */
public MatrixView(){
    matrix = null;           // keine Matrix
    pfade = new ArrayList(); // Liste anlegen
    breite = 10;           // Feldbreite
    hoehe = 10;           // Feldhöhe
    setSize(690,450);      // Standartgröße
    setPreferredSize(new Dimension(690,450)); // "

//////////
// Fenster zur Anzeige einiger statistischer Informationen
matrixInfo = new MatrixStatInfoDialog(matrix);
matrixInfo.setLocation(0,400);
matrixInfo.show();

//////////
// x,y-Koordinaten, Farbe und Wert werden in diesem Fenster angezeigt.
xyinfo = new XYInfo();
xyinfo.setLocation(0,100);
addMouseMotionListener(this); // Dazu benötigen wir die Mauskoordinaten
addMouseListener(this);

//////////
// SatzInfoDialog
satzinfo = new SatzInfoDialog(matrix);
satzinfo.setLocation(0,430);

//////////
// AbstandInfoDialog
abstandinfo = new AbstandInfoDialog(matrix);
abstandinfo.setLocation(0,500);

//////////
// Innere Klasse, die als TableModel dient, in gang bringen
tableModel= new T(this);

```

```

JDialog f = new JDialog((JFrame)null,"Mavis - Alignments");

// Dieser Dialog soll nicht schliessbar sein...
f.setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);

table = new JTable(tableModel);
table.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

    //Set up renderer and editor for the Favorite Color column.
    tableModel.setUpColorRenderer(table);
tableModel.setUpColorEditor(table);

// Spaltenbreiten setzen
TableColumn column = null;
column = table.getColumnModel().getColumn(0);
column.setPreferredWidth(25);
column = table.getColumnModel().getColumn(1);
column.setPreferredWidth(25);

JScrollPane scrollPane = new JScrollPane(table);
table.setPreferredScrollableViewportSize(new Dimension(220, 70));

f.getContentPane().add(scrollPane);
f.setLocation(0,200);
f.pack();
f.show();
//
//////////
}

/**
 * legt neue MatrixView mit Inhalt an
 * @param m Matrix, die angezeigt werden soll
 */
public MatrixView(Matrix m){
    // Defaultkonstruktor laufen lassen
    this();
    // Matrix merken
    matrix = m;
    // Wir sind an allen Änderungen an der Matrix interessiert!
    m.beobachterAnmelden(this);
    // bevorzugte Bildschirmabmessungen setzen
    setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));
    //setPreferredSize(new Dimension(640,400));
    // neu zeichnen!
    repaint();
}

/**
 * neues Matrix Objekt einpflanzen
 */
public void setzeMatrix(Matrix m){
    if(matrix != null) matrix.beobachterAbmelden(this); // wir melden uns als Beobachter der alten Matrix
    matrix = m;
    if(matrix != null){
        matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
        // Maße setzen
        setSize(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe);

```



```

        setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));
    }

    // bei Gelegenheit müssen wir uns neu zeichnen
    repaint();

    // häßlich! matrixInfo bescheid sagen
    matrixInfo.setzeMatrix(m);
    // nochmal ganz häßlich:
    satzinfo.setzeMatrix(m);
    abstandinfo.setzeMatrix(m);
}

/**
 * einen Pfad in die Liste der Alignment-Pfade aufnehmen,
 * Info-String ist mitzuliefern!
 */
public void fuegeAlignmentPfadHinzu(Pfad p, String info){
    if(pfade==null) pfade = new ArrayList();

    // Pfad als Element verpacken
    Elem e = new Elem(p, true, new Color((int)(Math.random()*127)+128,15,255-(int)(Math.random()*127)) , info);

    // Element in die PfadListe aufnehmen
    pfade.add(e);

    // Tabelle anstupsen
    tableModel.updateTable();

    // Selber neu zeichnen
    repaint();
}

/**
 * den selektierten Pfad aus der Liste der Alignment-Pfade löschen
 */
public void entferneSelektiertenAlignmentPfad(){
    if(pfade!=null){
        int index = table.getSelectedRow();
        if(index != -1){
            pfade.remove(index);
            tableModel.updateTable();
            repaint();
        }
    }
}

/**
 * den selektierten Pfad aus der Liste der Alignment-Pfade als HTML anzeigen
 */
public void zeigeSelektiertenAlignmentPfadAlsHTML(){
    if(pfade!=null){
        int index = table.getSelectedRow();
        if(index != -1){

            HTMLKocher kocher = new HTMLKocher(); // erzeugt html
            Elem elem = (Elem)pfade.get(index); // selektiertes Element besorgen
            Object o = elem.object; // enthaltenes Objekt rausholen

```

```

        // Ist das Objekt in der Liste ein Pfad?
        // Dann konvertieren!
        if(o instanceof Pfad){
            KoksZuordnung kneu = new KoksZuordnung((Pfad) o, matrix.holeQuelle(), matrix.holeInfo());
            o = kneu;
        }

        // ist es (jetzt) eine KoksZuordnung?
        // Dann mittels ShowHTMLFrame Ausgeben!
        if(o instanceof KoksZuordnung){
            new ShowHTMLFrame("Alignment", ""+kocher.kocheTabelle((KoksZuordnung)o,satzinfo.getShowTagsObjekte()));
        }
    }
}

public boolean istEtwasSelektiert(){
    return (table.getSelectedRow() != -1);
}

/**
 * Liste der Alignment-Pfade komplett löschen
 */
public void loescheAllePfade(){
    pfade = null;
    repaint();
    // Tabelle anstupsen
    tableModel.updateTable();
}

/**
 * Schreibt den selektierten Pfad (so vorhanden) in die Datei file
 */
public void speichereSelektiertenAlignmentPfad(File file){

    int index = table.getSelectedRow();
    if(index != -1){
        Elem elem = (Elem)pfade.get(index);

        if(elem.object instanceof Pfad){
            Pfad pfad = (Pfad) elem.object;

            try{
                FileWriter fw = new FileWriter(file);
                fw.write(pfad.toWagnerString(matrix.holeQuelle(), matrix.holeInfo()+" :: Alignment erzeugt von "+matrix.holeInfo()));
                fw.close();
            }catch(IOException e){
                System.err.println("Fehler beim Schreiben der Datei: "+ e.getMessage());
            }
        }
    }
}

/**
 * KoksZuordnung in die Liste der Pfade aufnehmen

```

```

*/
public void fuegeKoksZuordnungEin(KoksZuordnung zugekokst, String info){
    if(zugekokst==null) return;
    if(pfade==null) pfade = new ArrayList();

    Elem e = new Elem(zugekokst, true, new Color(15,(int)(Math.random()*255),(int)(Math.random()*255)) , info);
    pfade.add(e);

    // Tabelle anstupsen
    tableModel.updateTable();
    repaint();
}

/**
 * In dieser Methode finden das Zeichnen der
 * Matrix und der Pfade statt.
 */
public void paint(Graphics g){

    // erstmal den Hintergrund leerputzen
    super.paintComponent(g); //paint background

    // gefüllte Matrixfelder zeichnen
    if(matrix != null){
        for(int i=0; i< matrix.holeBreite(); i++){
            for(int j=0; j<matrix.holeHoehe(); j++){
                g.setColor(getColor(matrix.holeDatumVonStelle(i,j)));
                g.fillRect(i*(breite), j*(hoehe), breite, hoehe);
            }
        }
    }

    // Diagonale zeichnen
    if(zeigeDiag && matrix != null){
        g.setColor(new Color(0,0,0));
        g.drawLine(0,0,matrix.holeBreite()*breite, matrix.holeHoehe()*hoehe);
    }

    // KoksZuordnung zeichnen falls vorhanden (z.B. um Church und Gale aus Datei anzuzeigen...)
    // Pfadliste abklappern und sichtbare KoksZuordnungen ausgeben...
    if(pfade != null){
        for(int p = 0; p < pfade.size(); p++){
            // Element besorgen
            Elem elem = (Elem)pfade.get(p);
            // Handelt es sich um eine sichtbare KoksZuordnung, dann zeichnen
            if( (elem.object instanceof KoksZuordnung) && elem.visible){
                g.setColor(elem.color);
                KoksZuordnung kokszu = (KoksZuordnung) elem.object;
                ArrayList kokskoord = kokszu.betroffeneKoordinaten();
                int x,y;
                for (int i = 0; i < kokskoord.size(); i+=2){
                    x = ((Integer)kokskoord.get(i)).intValue();
                    y = ((Integer)kokskoord.get(i+1)).intValue();

                    if( (x >= 0) && (x < matrix.holeBreite()) &&

```

```

                (y >= 0) && (y < matrix.holeHoehe()) ){
                g.drawRect(x*(breite)+1, y*(hoehe)+1, breite-2, hoehe-2);
            }
        }
    }
} // if(pfade!=null)

if(matrix == null){
    // keine Matrix, dann mit tollem Logo protzen!
    // Logo ggbf. laden
    if(hgBild == null){
        URL url = this.getClass().getResource("/de/denkselfbst/mavis/logo.gif");
        Toolkit toolkit = Toolkit.getDefaultToolkit();
        hgBild = toolkit.getImage(url);
    }

    int h = hgBild.getHeight(this);
    int w = hgBild.getWidth(this);

    g.drawImage(hgBild, 0, 0, new Color(0,0,0), this);
}

// Pfade kennzeichnen
if (pfade != null){ // es gibt mindestens einen Pfad

    // Pfade abarbeiten
    for (int n=0; n < pfade.size(); n++){

        // Elem besorgen
        Elem elem = (Elem)pfade.get(n);
        // Handelt es sich um einen sichtbaren Pfad, dann zeichnen
        if( (elem.object instanceof Pfad) && elem.visible){
            // Farbe setzen
            g.setColor(elem.color);

            // Pfad besorgen
            Pfad p = (Pfad)elem.object;
            // Iterator über die Knoten des Pfades besorgen
            Iterator it = p.iterator();

            int oldx=0; // Startpunkt des Linienzuges
            int oldy=0; // "

            int b=breite/2 ;//(n*2); // Mittelpunkt-Offset in die gezeichneten Matrixfelder
            int h=hoehe/2 +(n*2); // "

            while(it.hasNext()){
                Pfad.Koordinate k = (Pfad.Koordinate) it.next();
                g.drawLine((oldx*breite)+b,(oldy*hoehe)+h,(k.x*breite)+b,(k.y*hoehe)+h);
                oldx=k.x;
                oldy=k.y;
            }
        }
    }
}
}
}
}
}

```

```

/**
 * einen double Wert zwischen 0 und 1 gegen einen Grünton eintauschen: 0 = sehr helles Grün,
 * 1 = sehr dunkles Grün, man könnte auch Schwarz dazu sagen
 */
public Color getColor(double wert){
    //if(wert < 0.000000000000000000) return new Color(150,0,0);
    //if(wert > 1.000000000000000000) return new Color(0,0,150);
    float w = (float) wert;
    if (w>=1.0f) w=1.0f;
    if (wert > 1.0) return new Color(150,5,5);
    return new Color(0,1-w,0); //
}

/**
 * Blockgröße setzen (Zeichnen)
 */
public void setzeBlockGroesse(int breite, int hoehe){
    if (matrix == null) return; // Defensiv...

    this.breite = breite;
    this.hoehe = hoehe;

    // Maße setzen
    setSize(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe);
    setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));
    revalidate();

    repaint();
}

/**
 * Schalter: Diagonale zeichnen oder nicht? (Zeichnen)
 */
public void zeigeDiagonale(boolean z){
    zeigeDiag = z;
    repaint();
}

public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
    // (neue) Maße setzen
    setSize(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe);
    setPreferredSize(new Dimension(matrix.holeBreite()*breite,matrix.holeHoehe()*hoehe));

    revalidate();

    repaint();
}

////////////////////////////////////
////////////////////////////////////
/** MouseMotionListener und MouseListener */
//
public void mouseDragged(MouseEvent e){
    // nichts tun...
}

```

```

public void mouseMoved(MouseEvent e){

    if(matrix != null && xyinfo != null){

        int x=0 ,y=0;

        x+= e.getX();
        y+= e.getY();

        x = x/ breite;
        y = y/ hoehe;

        if(x >= matrix.holeBreite() || y >= matrix.holeHoehe() ){
            xyinfo.dead();
            if (satzinfo != null) satzinfo.dead();
            if (abstandinfo != null) abstandinfo.dead();
        }else{
            double data = matrix.holeDatumVonStelle(x,y);
            xyinfo.update(x,y,getColor(data),data);
            if(satzinfo!=null) satzinfo.show(x,y);
            if(abstandinfo!=null) abstandinfo.show(x,y);
        }
    }
}

public void mouseClicked(MouseEvent e){}
public void mouseEntered(MouseEvent e){}
public void mouseExited(MouseEvent e){//xyinfo.dead(); if(satzinfo!=null)satzinfo.dead();
}
public void mousePressed(MouseEvent e){}
public void mouseReleased(MouseEvent e){}

////////////////////////////////////
////////////////////////////////////
//
/** Memberklasse, die in einem eigenen Fenster
 * die aktuellen MatrixKoordinaten und den zugehörigen Wert anzeigt
 */
public class XYInfo extends JDialog {

    JLabel xp,yp,col,val;

    /** Konstruktor */
    public XYInfo(){
        super((JFrame)null,"Matrixzellen - Info");
        setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);

        xp = new JLabel("x"); //xp.setHorizontalAlignment(JLabel.RIGHT);
        yp = new JLabel("y");
        col = new JLabel("      ");
        col.setOpaque(true);
        val = new JLabel("      ");
        JPanel p = new JPanel(new GridLayout(4,2));

        p.add(new JLabel("X")); p.add(xp);
        p.add(new JLabel("Y")); p.add(yp);
    }
}

```

```

p.add(new JLabel("Farbe")); p.add(col);
p.add(new JLabel("Wert")); p.add(val);

getContentPane().add(p);
setSize(new Dimension(220,70));
show();

}

/** Die neuen Werte zur Anzeige bringen */
public void update(int x, int y, Color color, double value){
    xp.setText(""+x);
    yp.setText(""+y);
    col.setBackground(color);
    col.setForeground(color);
    val.setText(""+value);
    //repaint(); // verursachen die Labels das von selbst?
}

/** Teilt diesem Objekt mit, dass keine anzeigbaren Werte vorliegen.
 * z.B. Maus ausserhalb des Fensters
 */
public void dead(){
    xp.setText("-");
    yp.setText("-");
    val.setText("-");
    col.setBackground(Color.lightGray);
    col.setForeground(Color.lightGray);
}

}

////////////////////////////////////
////////////////////////////////////
//
/** eklige Memberklasse, die als Tablemodel dient */
//

public class T extends AbstractTableModel{

    String[] columnNames = new String[]{"sichtbar","Farbe","Info","Ende der Durchsage"};
    private MatrixView mv;

    public T(MatrixView mv){
        this.mv = mv;
    }

    public void updateTable(){
        fireTableDataChanged();
    }

    public int getColumnCount(){
        return 3;
    }

    public int getRowCount(){
        if(pfade!=null) return pfade.size(); // aus der umgebenden Klasse die Arrayliste Pfade
        else return 0;
    }
}

```

```

}

public String getColumnName(int col) {
    return columnNames[col];
}

public Class getColumnClass(int c) {
    //return getValueAt(0, c).getClass();

    if(c==0) return new Boolean(true).getClass();
    if(c==1) return new Color(1,2,3).getClass();
    if(c==2) return new String("").getClass();

    return null;
}

public Object getValueAt(int row, int column){
    if(row < pfade.size()){
        Elem elem = (Elem) pfade.get(row);
        if(column==0) return new Boolean(elem.visible);
        else if(column==1) return elem.color;
        else if(column==2) return elem.info;
    }
    else{
        if(column==0) return new Boolean(false);
        if(column==1) return new Color(1,2,3);
        if(column==2) return new String("dummy");
    }
    return new Integer(666);
}

public boolean isCellEditable(int row, int col) {
    if ( (col >= 0) && (col <=1) )
        return true;
    return false;
}

public void setValueAt(Object value, int row, int col) {
    Elem elem = (Elem)pfade.get(row);

    if(col==0){
        elem.visible=((Boolean)value).booleanValue();
    }
    else if(col==1){
        elem.color=(Color)value;
    }

    fireTableCellUpdated(row, col);
    mv.repaint();
}

////////////////////////////////////
/* noch eine innere Klasse, diesmal, um
 * Farben in der Pfadliste darzustellen und
 * Farbauswahl zu ermöglichen. Quelle: Java-Tutorial von Sun.
 */

class ColorRenderer extends JLabel
    implements TableCellRenderer {

```



```

Border unselectedBorder = null;
Border selectedBorder = null;
boolean isBordered = true;

public ColorRenderer(boolean isBordered) {
    super();
    this.isBordered = isBordered;
    setOpaque(true); //MUST do this for background to show up.
}

public Component getTableCellRendererComponent(
    JTable table, Object color,
    boolean isSelected, boolean hasFocus,
    int row, int column) {
    setBackground((Color)color);
    if (isBordered) {
        if (isSelected) {
            if (selectedBorder == null) {
                selectedBorder = BorderFactory.createMatteBorder(2,5,2,5,
                    table.getSelectionBackground());
            }
            setBorder(selectedBorder);
        } else {
            if (unselectedBorder == null) {
                unselectedBorder = BorderFactory.createMatteBorder(2,5,2,5,
                    table.getBackground());
            }
            setBorder(unselectedBorder);
        }
    }
    return this;
}

private void setUpColorRenderer(JTable table) {
    table.setDefaultRenderer(Color.class,
        new ColorRenderer(true));
} //Set up the editor for the Color cells.

private void setUpColorEditor(JTable table) {
    //First, set up the button that brings up the dialog.
    final JButton button = new JButton("") {
        public void setText(String s) {
            //Button never shows text -- only color.
        }
    };
    button.setBackground(Color.white);
    button.setBorderPainted(false);
    button.setMargin(new Insets(0,0,0,0)); //Now create an editor to encapsulate the button, and
    //set it up as the editor for all Color cells.
    final ColorEditor colorEditor = new ColorEditor(button);
    table.setDefaultEditor(Color.class, colorEditor); //Set up the dialog that the button brings up.
    final JColorChooser colorChooser = new JColorChooser();
    ActionListener okListener = new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            colorEditor.currentColor = colorChooser.getColor();
        }
    };
    final JDialog dialog = JColorChooser.createDialog(button,
        "Pick a Color",
        true,

```

```

        colorChooser,
        okListener,
        null); //XXXDoublecheck this is OK //Here's the code that bri
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        button.setBackground(colorEditor.currentColor);
        colorChooser.setColor(colorEditor.currentColor);
        //Without the following line, the dialog comes up
        //in the middle of the screen.
        //dialog.setLocationRelativeTo(button);
        dialog.show();
    }
});
} /*
 * The editor button that brings up the dialog.
 * We extend DefaultCellEditor for convenience,
 * even though it mean we have to create a dummy
 * check box. Another approach would be to copy
 * the implementation of TableCellEditor methods
 * from the source code for DefaultCellEditor.
 */
class ColorEditor extends DefaultCellEditor {
    Color currentColor = null;    public ColorEditor(JButton b) {
        super(new JCheckBox()); //Unfortunately, the constructor
                                //expects a check box, combo box,
                                //or text field.

        editorComponent = b;
        setClickCountToStart(1); //This is usually 1 or 2. //Must do this so that editing sto
        b.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                fireEditingStopped();
            }
        });
    }    protected void fireEditingStopped() {
        super.fireEditingStopped();
    }    public Object getCellEditorValue() {
        return currentColor;
    }    public Component getTableCellEditorComponent(JTable table,
                                                    Object value,
                                                    boolean isSelected,
                                                    int row,
                                                    int column) {
        ((JButton)editorComponent).setText(value.toString());
        currentColor = (Color)value;
        return editorComponent;
    }
}
}
}

```

M.5.5 de.denkselbst.matrix.MatrixHistogramView.java

```

package de.denkselbst.matrix;

import de.denkselbst.beobachtung.Beobachter;

import javax.swing.JPanel;

```

```

import java.awt.Color;
import java.awt.Graphics;

/**
 * Diese Klasse ist eine grafische Komponente, die in einem eigenen
 * Fenster ein Histogramm eines dem Konstruktor übergebenen
 * Matrix-Objektes zeichnet.
 */

public class MatrixHistogramView extends JPanel implements Beobachter {

    /** Referenz auf die als Histogramm darzustellende Matrix */
    private Matrix matrix;
    private int[] histogramm;
    private double schrittGroesse;

    int viewbreite = 180;
    int viewhoehe = 70;
    float scale = 1.0f;    // wird benutzt, um die Balken immer schön groß und deutlich zu zeichnen...

    /** Konstruktor */
    public MatrixHistogramView(Matrix m){
        matrix = m;
        schrittGroesse = 0.05;
        // Wir sind an allen Änderungen an der Matrix interessiert!
        matrix.beobachterAnmelden(this);
        histogramm = berechneHistogramm(schrittGroesse);
        computeScale();
    }

    /** Das tatsächliche Zeichnen findet hier statt. */
    public void paint(Graphics g){
        // Anzahl Matrixfelder berechnen
        int groesse = matrix.holeHoehe() * matrix.holeBreite();

        int breite = viewbreite - 20 - (histogramm.length - 1); // Gesamtbreite für alle Balken zusammen
        int balkenBreite = breite / histogramm.length ;           // Breite eines Balkens

        // erstmal den Hintergrund leerputzen
        super.paintComponent(g); //paint background

        for(int i = 0; i < histogramm.length; i++){
            float col = (float) schrittGroesse * (i+1);
            if (col < 0 || col > 1){
                g.setColor(new Color(175,10,10));
            }
            else g.setColor(new Color(0,1-col,0));

            int laenge = (int) (scale * (( 1.0 * histogramm[i]) / groesse) * viewhoehe));
            g.fillRect(10+i* balkenBreite, viewhoehe- laenge, balkenBreite, laenge);
        }
    }

    /** Histogramm / Balkenhöhen /-Anzahlen ausrechnen
     * und Balkenhöhenarray liefern.
     */
    private synchronized int[] berechneHistogramm (double stepsize){
        int hoehe = matrix.holeHoehe();
        int breite = matrix.holeBreite();

```

```

int anzahlBalken = (int)Math.ceil(1.0 / stepsize);
int [] balken = new int[anzahlBalken];

for(int i = 0; i < breite; i++){
    for(int j = 0; j < hoehe; j++){

        double wert = matrix.holeDatumVonStelle(i,j);
        for(int z = 0; z < anzahlBalken; z++){
            if (wert >= z * stepsize && wert < (z+1)*stepsize){
                balken[z]++;
                continue;
            }
        }
    }
}
return balken;
}

/** Berechnet internen Vergrößerungsfaktor, damit das
    * Histogramm immer schön sichtbar ist.
    */
private void computeScale(){
    if(histogramm == null) return;

    int groesse = matrix.holeHoehe() * matrix.holeBreite();
    int max = 0;
    for(int i=0; i<histogramm.length; i++){
        if (histogramm[i] > max) max = histogramm[i];
    }

    if(max ==0) max = 1;

    float relmax = (float) (1.0f*max / groesse);
    scale = 1.0f / relmax;

    if (scale == 0) scale = 1.0f;
}

/** Schrittgröße verändern*/
public void setzeSchrittgroesse(double g){
    schrittGroesse = g;
    histogramm = berechneHistogramm(schrittGroesse);
    computeScale();
    repaint();
}

/** Beobachter: Ein update der Anzeige wird nötig. */
public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
    histogramm = berechneHistogramm(schrittGroesse);
    computeScale();
    repaint();
}

/** Die Beobachtung des Matrix-Objektes einstellen */
public void abmelden(){
    matrix.beobachterAbmelden(this);
}

```

```

}
}

```

M.5.6 de.denkselbst.matrix.Matrix.MatrixStatInfoDialog.java

```

package de.denkselbst.matrix;

import de.denkselbst.beobachtung.Beobachter;
import javax.swing.*.*;
import java.awt.*.*;

/** Diese Klasse zeigt in einem eigenen Fenster
 *?Dateiname, Mittelwert, Varianz, Standardabweichung der
 * dem Konstruktor übergebenen Matrix an.
 */
// Alle diese Views bei Gelegenheit ausgliedern!
// Vielleicht im nächsten Projekt.

public class MatrixStatInfoDialog extends JDialog implements Beobachter{

    Matrix matrix;
    JLabel dateiname, info;
    JLabel mittelwert, varianz, standardabweichung;
    JLabel dimensionen, minimum, maximum;

    /** Konstruktor */
    public MatrixStatInfoDialog(Matrix mat){
        super((JFrame)null, "Matrixinfo");
        setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);

        this.setzeMatrix(mat);

        JLabel dn = new JLabel("Quelle");
        JLabel nf = new JLabel("Info");
        JLabel di = new JLabel("x * y");
        JLabel mi = new JLabel("Mittelw.");
        JLabel va = new JLabel("Varianz");
        JLabel st = new JLabel("Stdabw.");
        JLabel min = new JLabel("Minimum");
        JLabel max = new JLabel("Maximum");

        dateiname = new JLabel("...");
        info = new JLabel("...");
        mittelwert = new JLabel("...");
        varianz = new JLabel("...");
        standardabweichung = new JLabel("...");
        dimensionen = new JLabel("...");
        minimum = new JLabel("...");
        maximum = new JLabel("...");

        JPanel p = new JPanel(new GridLayout(8,2));

        p.add(dn); p.add(dateiname);
        p.add(nf); p.add(info);
        p.add(di); p.add(dimensionen);
        p.add(mi); p.add(mittelwert);
        p.add(va); p.add(varianz);
        p.add(st); p.add(standardabweichung);

```

```

    p.add(min); p.add(minimum);
    p.add(max); p.add(maximum);

    getContentPane().add(p);
    setSize(new Dimension(220,100));

    update();

    show();
}

/** Wird durch Observermuster angestossen, sobald
    * das Matrixobjekt sich ändert. Sorgt dafür, das
    * die angezeigten Werte aktualisiert werden.
    */
private void update(){
    if(matrix != null){
        dateiname.setText(matrix.holeQuelle());
        info.setText(matrix.holeInfo());
        dimensionen.setText(""+matrix.holeBreite()+" x "+matrix.holeHoehe());
        mittelwert.setText(""+matrix.berechneMittelwert());
        varianz.setText(""+matrix.berechneVarianz());
        standardabweichung.setText(""+matrix.berechneStandardabweichung());
        minimum.setText(""+matrix.wasIstDerKleinsteWert());
        maximum.setText(""+matrix.wasIstDerGrossteWert());
    }
}

/**
 * neues Matrix Objekt einpflanzen
 */
public void setzeMatrix(Matrix m){
    if(matrix != null) matrix.beobachterAbmelden(this); // wir melden uns als Beobachter der alten Mat
    matrix = m;
    if(matrix != null){
        matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
        update();
    }
}

public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
    update();
}
}

```

M.5.7 de.denkselbst.matrix.Matrix.SatzInfoDialog.java

```

package de.denkselbst.matrix;

//import de.denkselbst.matrix.Matrix;

import de.denkselbst.beobachtung.Beoachter;
import javax.swing.*.*;
import java.awt.*.*;

/**
 * Diese Klasse zeigt in einem Fenster Sätze textuell an, die zu der
 * aktuell selektierten Matrixzelle gehören.

```

```

* Dazu wird dem Konstruktor ein Matrix-Objekt übergeben. SatzInfoDialog
* meldet sich bei der Matrix als Beobachter an und besorgt sich die Quell-
* Information derselben. Es wird ein ShowTags-Objekt angelegt, das mittels
* der Quelleninformation möglicherweise die entsprechenden Textdateien
* lesen kann.
* Das MatrixView-Objekt reagiert auf Mausbewegungen und weist sein
* SatzInfoDialog-Objekt an, die zur aktuell unter dem Mauscursor
* befindlichen Matrixzelle gehörenden Sätze anzuzeigen.
*/
public class SatzInfoDialog extends JDialog implements Beobachter{

    private JTextArea de, en;
    private ShowTags textquelle;
    private Matrix matrix;

    /** Konstruktor */
    public SatzInfoDialog(Matrix mat){
        super((JFrame)null,"Satzinfo");
        setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
        matrix = mat;
        de = new JTextArea("deutscher Text"/*,8,35*/);
        en = new JTextArea("englischer Text"/*,8,35*/);
        de.setLineWrap(true); de.setWrapStyleWord(true);
        en.setLineWrap(true); en.setWrapStyleWord(true);
        JPanel p = new JPanel(new GridLayout(1,2));
        JScrollPane ensp = new JScrollPane(en, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        en.setPreferredSize(new Dimension(320,100));
        JScrollPane desp = new JScrollPane(de, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        de.setPreferredSize(new Dimension(320,100));
        p.add(ensp);
        p.add(desp);
        getContentPane().add(p);
        setSize(640,120);
        updateQuelle();
        pack();
        show();
    }

    /** Liefert benutztes ShowTags-Objekt */
    public ShowTags getShowTagsObject(){
        return textquelle;
    }

    /** Erzeugt ShowTags-Objekt für internen Gebrauch */
    public void updateQuelle(){
        if(matrix != null)
            textquelle = new ShowTags(matrix.holeQuelle());
        else textquelle = null;
    }

    /** Sorgt dafür, das die Sätze x und y textuell im
        * Fenster angezeigt werden
        */
    public void show(int x, int y){
        if(textquelle == null) return;
        de.setText(textquelle.getY(y));
        en.setText(textquelle.getX(x));
    }

    /** Wird aufgerufen, wenn der Mauszeiger nicht über
        * einer Matrixzelle steht. In diesem Fall kann nichts

```

```

        * angezeigt werden.
        */
    public void dead(){
        de.setText("...");
        en.setText("...");
    }

    /**
     * neues Matrix Objekt einpflanzen
     */
    public void setzeMatrix(Matrix m){
        if(matrix != null) matrix.beobachterAbmelden(this); // wir melden uns als Beobachter der alten Mat
        matrix = m;
        if(matrix != null){
            matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
            updateQuelle();
        }
    }

    public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
        updateQuelle();
    }

}

```

M.5.8 de.denkselbst.matrix.Matrix.ShowTags.java

```

package de.denkselbst.matrix;

import java.io.*;
import java.util.ArrayList;

/**
 * Diese Klasse besorgt, falls möglich, die zu einer Matrix
 * gehörenden Texte und liefert auf Anfrage Sätze (Strings) zu
 * übergebenen Satznummern (int).
 */
public class ShowTags{

    private ArrayList deutsch;
    private ArrayList englisch;

    /** Konstruktor */
    public ShowTags(String quelle){
        //nur wenn quelle das Format: "Dokument-Datei ../korpus/de-news/1997/05/de-news-1997-05-03"
        //hat, können wir etwas damit anfangen!
        String dateistamm = null;
        if(quelle.startsWith("Dokument-Datei")){ // Dann brauchbar!
            dateistamm = quelle.substring(15);
        }

        deutsch = new ArrayList();
        englisch = new ArrayList();

        fillFromFile(deutsch,dateistamm+".de.tag");
        fillFromFile(englisch,dateistamm+".en.tag");
    }
}

```



```

System.err.println("de: "+deutsch.size()+" Sätze");
System.err.println("en: "+englisch.size()+" Sätze");
}

// x englisch
// y deutsch
/** Liefert Satz x aus L1 als String. */
public String getX(int x){
    if (x >= 0 && x < englisch.size()) return ((String)englisch.get(x)).trim();
    else return "keine Daten an x="+x;
}
/** Liefert Satz y aus L2 als String. */
public String getY(int y){
    if (y >= 0 && y < deutsch.size()) return ((String)deutsch.get(y)).trim();
    else return "keine Daten an y="+y;
}

/* Liefert die Anzahl der Sätze aus L1.*/
public int xmax(){
    return englisch.size();
}
/* Liefert die Anzahl der Sätze aus L2.*/
public int ymax(){
    return deutsch.size();
}

/**
 * Füllt übergebene ArrayList <SATZ>-Weise aus Datei dateiname
 */
private void fillFromFile(ArrayList list, String dateiname) {

    // Nicht gerade der beste Weg um herauszufinden, dass kein vernünftiger Dateiname!
    if(dateiname.equals("null.de.tag") || dateiname.equals("null.en.tag")){
        System.err.println("diese Quelle kann ich nicht anzeigen");
        return;
    }

    //System.err.println("***** Showtags: "+dateiname);

    try{
        String line = null;
        StringBuffer akkumulator = new StringBuffer();
        String wort = null;

        int zustand = 0;

        File file = new File(dateiname);
        BufferedReader br = new BufferedReader(new FileReader(file));

        while ( (line=br.readLine()) != null ){

            int pos = line.indexOf("\t");
            if(pos >= 0 && pos < line.length()) wort = line.substring(0,pos).trim();
            else wort = line;
            //System.out.println(wort);

            //          (pos >=0 : Indikator dafür, dass es sich um Inhalt handelt)
            if(zustand == 0 && pos > 0 ){
                //System.err.println("[1] akk: "+akkumulator.toString()+" wort: "+wort);

```



```

/**
 * Diese Klasse enthält eine statische Methode, die
 * KoksZuordnung unter Zuhilfenahme eines ShowTags-Objektes
 * und diversen info-Strings ls HTML-String liefert.
 * So kann ein Alignment in einer HTML-View textuell
 * dargestellt werden.
 */

public class HTMLKocher{

    /**
     * Diese statische Methode gibt eine KoksZuordnung @param k
     * unter Zuhilfenahme eines ShowTags-Objektes @param s
     * und diversen info-Strings
     * als HTML-String zurück.
     * @param info Info-String des Aligners / Matrix (z.B. AStern, Lmed)
     * @param parameter String, der die gesetzten Parameter des Aligners beschreiben soll (in Zukunft)
     * @param quelle Herkunft der Daten (analog zu Matrix.quelle)
     *
     */
    public static String kocheTabelle(KoksZuordnung k, ShowTags s, String alinfo, String parameter){

        String info = k.holeInfo();
        String quelle = k.holeQuelle();

        // alternierende Hintergrundfarben für die Tabelle
        String[] colors = new String[]{"BGOLOR=#FFFFFFCC\""}, "BGOLOR=#FFFFFFE4\""};
        int currentColorIndex = 0;
        String currentColor;

        ArrayList zuordnungen = k.getZuordnungen();

        StringBuffer buf = new StringBuffer("<html>\n <head></head>\n <body>");
        // Später noch alle verfügbaren Infos zum verwendeten Aligner
        // (Art, Parameter!)
        // und ein gif der Tabelle
        // ausgeben, damit Ausdrücke besser verglichen werden können.

        buf.append("\n Datum: "+new Date(System.currentTimeMillis())+" <br> Quelle: "+quelle+" <br> Info:"+info+"<br>");
        buf.append("<h1>Alignment: "+alinfo+"</h1>");
        buf.append("\n <table border cols = 2>");

        // Hintergrundfarbe für Tabellenheader-Zeile besorgen
        currentColor = colors[currentColorIndex];
        // nächste Farbe (Index ins Array bestimmen)
        currentColorIndex = (currentColorIndex + 1) % colors.length;
        // HTML-Tabelle: Headerzeile ausgeben
        buf.append("\n <tr "+currentColor+" ><th>L1</th><th>L2</th></tr>");

        // Alle Zuordnungen abarbeiten und ausgeben
        for(int i = 0; i < zuordnungen.size(); i++){
            KoksZuordnung.Zuordnung z = (KoksZuordnung.Zuordnung)zuordnungen.get(i);

            // z ist ein Segment
            ArrayList lang1 = z.getL1(); // Sprache 1 (Satz)Nummern
            ArrayList lang2 = z.getL2(); // Sprache 2 (Satz)Nummern

            // Hintergrundfarbe für die Tabellenzeile besorgen
            currentColor = colors[currentColorIndex];

            // HTML-Tabelle: Zeile mit Hintergrundfarbe öffnen

```

```

buf.append("\n <tr "+currentColor+" >\n <td valign=top>");

// Sprache 1 behandeln:
// Segmentnummer und Text ausgeben
for(int j = 0; j < lang1.size(); j++){
    buf.append("[ "+((Integer)lang1.get(j)).intValue()+"]"); // Segmentnummer
    buf.append( s.getY(((Integer)lang1.get(j)).intValue()) ); // Text
    buf.append("<br>\n"); // Linebreak
}

// HTML-Tabelle: nächste Spalte aufmachen
buf.append(" </td>\n <td valign=top>");

// Sprache 2 behandeln
// Segmentnummer und Text ausgeben
for(int j = 0; j < lang2.size(); j++){
    buf.append("[ "+((Integer)lang2.get(j)).intValue()+"]"); // Segmentnummer
    buf.append( s.getX(((Integer)lang2.get(j)).intValue()) ); // Text
    buf.append("<br>\n"); // Linebreak
}

// HTML-Tabelle: Spalte, Zeile schließen
buf.append("</td>\n </tr>");

// nächste Farbe (Index ins Array bestimmen)
currentColorIndex = (currentColorIndex + 1) % colors.length;

}
buf.append("\n </table>\n </body>\n</html>");

return buf.toString();
}
}

```

M.5.10 de.denkselbst.matrix.Matrix.AbstandInfoDialog.java

```

package de.denkselbst.matrix;

//import de.denkselbst.matrix.Matrix;

import de.denkselbst.beobachtung.Beobachter;
import javax.swing.*;
import java.awt.*;

/**
 * Nach dem Vorbild von SatzInfoDialog geklonte Klasse zum
 * Anzeigen von Debugging-Informationen für Abstandsmaße.
 * Autor: Joachim.
 */

public class AbstandInfoDialog extends JDialog implements Beobachter{

    private JTextArea de, en;
    private ShowAbstand textquelle;
    private Matrix matrix;

    public AbstandInfoDialog(Matrix mat){
        super((JFrame)null, "Abstandinfo");
        setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
    }
}

```

```

matrix = mat;
de = new JTextArea("Info 1"/*,8,35*/);

de.setLineWrap(true); de.setWrapStyleWord(true);

JPanel p = new JPanel(new GridLayout(1,1));

JScrollPane desp = new JScrollPane(de, JScrollPane.VERTICAL_SCROLLBAR_ALWAYS, JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
de.setPreferredSize(new Dimension(320,100));

p.add(desp);

getContentPane().add(p);

setSize(640,120);

//setLocation(200,550);

updateQuelle();

pack();
show();
}

public ShowAbstand getShowAbstandObject(){
    return textquelle;
}

public void updateQuelle(){
    if(matrix != null)
        textquelle = new ShowAbstand(matrix.holeQuelle());
    else textquelle = null;
}

public void show(int x, int y){
    if(textquelle == null) return;
    de.setText(textquelle.getAbstandinfo(x,y));
}

public void dead(){
    de.setText("...");
}

/**
 * neues Matrix Objekt einpflanzen
 */
public void setzeMatrix(Matrix m){
    if(matrix != null) matrix.beobachterAbmelden(this); // wir melden uns als Beobachter der alten Matrix
    matrix = m;
    if(matrix != null){
        matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
        updateQuelle();
    }
}

public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
    //System.out.println("..." +nachricht);
    updateQuelle();
}

```

```

    }

    public static void main(String[] args){
        AbstandInfoDialog s = new AbstandInfoDialog(null);
    }
}

```

M.5.11 de.denkselbst.matrix.Matrix.ShowAbstand.java

```

package de.denkselbst.matrix;

import java.io.*;
import java.util.ArrayList;
import java.util.StringTokenizer;

/**
 * Nach dem Vorbild von ShowTags geklonte Klasse zum
 * Anzeigen von Debugging-Informationen für Abstandsmaße.
 * Autor: Joachim.
 */

public class ShowAbstand{

    private ArrayList abstandinfo;

    private int breite, hoehe;

    /** Konstruktor */
    public ShowAbstand(String quelle){

        //nur wenn quelle das Format: "Dokument-Datei ../korpus/de-news/1997/05/de-news-1997-05-03"
        //hat, können wir etwas damit anfangen!
        String dateistamm = null;
        if(quelle.startsWith("Dokument-Datei")){ // Dann brauchbar!
            dateistamm = quelle.substring(15);
        }

        abstandinfo = new ArrayList();

        //System.err.println("dateistamm: "+dateistamm);

        fillFromFile(abstandinfo,dateistamm+".abstandinfo");

        System.err.println(": "+abstandinfo.size()+" Abstandinfos");
    }

    /**
     *
     */
    for(int i = abstandinfo.size()-5; i < abstandinfo.size(); i++){
        System.err.println("de:"+i+": "+getAbstandinfo(i));
    }
    System.err.println();
}

```

```
*/

}

// x englisch
// y deutsch

public String getAbstandinfo(int x, int ny){
    int y = breite * ny + x;
    if (y >= 0 && y < abstandinfo.size()) return ((String)abstandinfo.get(y)).trim();
    else return "keine Daten an x,y = "+x+", "+ny;
}

public int xmax(){
    return breite * hoehe;
}

public int ymax(){
    return breite * hoehe;
}

/**
 * Füllt übergebene ArrayList <SATZ>-Weise aus Datei dateiname
 */
private void fillFromFile(ArrayList list, String dateiname) {

    // Nicht gerade der beste Weg um herauszufinden, dass kein vernünftiger Dateiname!
    if(dateiname.equals("null.de.abstandsinfo") || dateiname.equals("null.en.abstandsinfo")){
        System.err.println("diese Quelle kann ich nicht anzeigen");
        return;
    }

    try{
        String line = null;
        StringBuffer akkumulator = new StringBuffer();

        int zustand = 0;

        File file = new File(dateiname);
        BufferedReader br = new BufferedReader(new FileReader(file));

        line = br.readLine();
        StringTokenizer tok = new StringTokenizer(line, "\t \n", false);

        // nächstes Token besorgen
        if(!tok.hasMoreTokens()){
            while(!tok.hasMoreTokens() && line!=null){
                line = br.readLine();
                tok = new StringTokenizer(line, "\t \n", false);
            }
        }
        // breite der Matrix holen
        breite = Integer.parseInt(tok.nextToken());

        // nächstes Token besorgen
        if(!tok.hasMoreTokens()){
            while(!tok.hasMoreTokens() && line!=null){
```



```

System.out.println();

*/

// Teste ShowAbstand in Verbindung mit HTMLKocher

String x = args[0].substring(15);
System.err.println("öffne: "+x+".align");
KoksZuordnung k = new KoksZuordnung(x+".align");

// HTMLKocher kocher = new HTMLKocher();
// System.out.println(""+kocher.kocheTabelle(k,st,"fakealigner","keine Parameter")); // st kein ShowTags

}

}

```

M.5.12 de.denkselbst.matrix.Matrix.WriteRandomMatrix.java

```

package de.denkselbst.matrix;

/**
 * Write random matrix to stdout. Values are between 0.0 and 1.0. First command line
 * argument is taken as xdim, second as ydim. Default dims: 10 x 10
 * Data is written row per row from left to right.
 * The output has the following format: xdim \t ydim \t vall \t ... \t valn
 * and can be read by Matrix-objects.
 * The aim is to provide test material to Matrix-objects.
 */
public class WriteRandomMatrix{
    public static void main (String[] args){
        int x = 10; // default dimensions
        int y = 10; // "

        if (args.length > 0)
            x = Integer.parseInt(args[0]);
        if (args.length > 1)
            y = Integer.parseInt(args[1]);

        System.out.println("random matrix");
        System.out.println("fuer Testzwecke");
        System.out.print(""+x+"\t"+y);

        for(int i = 0; i < x; i++){
            for(int j = 0; j < y; j++){
                System.out.print("\t"+Math.random());
            }
        }
    }
}

```

M.5.13 de.denkselbst.matrix.Matrix.Convert.java

```

package de.denkselbst.matrix;

import java.util.ArrayList;
import java.util.StringTokenizer;
import java.io.BufferedReader;

```

```

import java.io.IOException;
import java.io.FileReader;
import java.io.File;
import java.io.FileWriter;

/**
 * Die Main-Methode dieser Klasse veranlasst, das eine zeilenweise
 * eingelesene Matrix spaltenweise ausgegeben wird.
 * inputdatei = argument[n], outputdateiname = inputdateiname+".converted"
 *
 * Da es nach einigen Anlaufschwierigkeiten mit dem Matrixformat keine
 * Probleme mehr gab, war es schon früh nicht mehr nötig, Matrizen mit
 * diesem Programm zu drehen.
 *
 * !!Diese Klasse ist daher obsolet!!
 * Allerdings ist sie noch immer Teil des CVS-Baumes. Der Code zum
 * Einlesen einer Matrix wurde hier dupliziert,
 * was eine Sünde ist.
 */
public class Convert{
    public static void main (String[] args) throws IOException {
        for(int a = 0 ; a < args.length; a++){
            double m[][] = readFromFile(args[a]);
            int x = m.length;
            int y = m[0].length;
            File outputFile = new File(args[a]+".converted");
            FileWriter out = new FileWriter(outputFile);

            out.write(""+x+"\t"+y+"\t");
            for(int i = 0 ; i < x; i++){
                for(int j = 0; j < y; j++){
                    out.write(""+m[i][j)+"\t");
                }
            }
            out.close();
        }
    }

    /**
     * liest Daten aus einer Datei ein und liefert double[][]
     * @param file Dateiname als String
     */
    private static double[][] readFromFile(String file){
        int x = 0; // Abmessungen der Matrix
        int y = 0; // "
        double[][] matrix = new double[0][0]; // Referenz auf Matrix

        try{
            BufferedReader in = new BufferedReader(new FileReader(file));
            String line = in.readLine();
            StringTokenizer tok = new StringTokenizer(line, "\t\n", false);

            // nächstes Token besorgen
            if(!tok.hasMoreTokens()){
                while(!tok.hasMoreTokens() && line!=null){
                    line = in.readLine();
                    tok = new StringTokenizer(line, "\t\n", false);
                }
            }
            // breite der Matrix holen

```

```

x = Integer.parseInt(tok.nextToken());

// nächstes Token besorgen
if(!tok.hasMoreTokens()){
    while(!tok.hasMoreTokens() && line!=null){
        line = in.readLine();
        tok = new StringTokenizer(line, "\t \n", false);
    }
}
// Höhe der Matrix holen
y = Integer.parseInt(tok.nextToken());

// Matrix anlegen
matrix = new double[x][y];
String token = null;

// Zeilenweise einlesen
for (int j = 0; j < y; j++){
    for (int i = 0 ; i < x ; i++){
        // Nächstes Token besorgen
        if(!tok.hasMoreTokens()){
            while(!tok.hasMoreTokens() && line!=null){
                line = in.readLine();
                tok = new StringTokenizer(line, "\t \n", false);
            }
        }
        token = tok.nextToken();
        matrix[i][j] = Double.parseDouble(token);
    }
}
} catch(IOException e){
    System.err.println(e.getMessage());
    System.err.println("Fehler beim Lesen der Datei "+file+" . Liefere leere 6x6 Matrix.");
    return new double[6][6];
}

return matrix; // Die Matrix liefern
}

}

```

M.5.14 de.denkselbst.matrixAligner.MatrixAligner.java

```

package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.beobachtung.Beobachter;

/**
Diese Klasse ist Superklasse aller Klassen, die einen Alignment-Pfad für eine Matrix
ermitteln wollen, können oder sollen.

```

Die Klasse MatrixAligner schreibt vor, wie sich Objekte zu verhalten haben, die einen Alignmentpfad in einer Matrix ermitteln wollen. Für das GUI-Tool Mavis wurden Methoden bereitgestellt, das Alignment in einem separaten Thread auszuführen. Für Kommandozeilentools wird diese Funktionalität nicht benötigt. Alignments werden in dieser Version als Pfad-Objekte abgeliefert. Später in

der Entwicklung wurde die Klasse KoksZuordnung geschaffen, die Alignments adäquater repräsentiert. Pfade lassen sich in jedem Fall in KoksZuordnungen umwandeln.

Soll das Alignment in einem eigenen Thread ablaufen, so ist die Methode `public void align(AlignmentObserver obs)` aufzurufen. Dabei ist zu beachten, dass dieser Aufruf pro Alignerobjekt nur einmal stattfinden darf, da sonst der zweite Aufruf den ersten Observer überschreibt. Also: Alignerobjekt instantiieren, einmal alignen, dann Alignerobjekt wegwerfen. Anschließend neues Alignerobjekt machen, alignen, wegwerfen, usw.

```

*/

public abstract class MatrixAligner implements Beobachter, AlignmentObservable, Runnable{
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
// Konstruktor, Matrix merken, Auskunft über alignerIdentString geben, Konfigurationsdialog aufrufen
//
// enthält Referenz auf die Matrix */
Matrix matrix;
// String, der den Typ dieses Aligners beinhaltet (muss/sollte in den Unterklassen gesetzt werden) */
String alignerIdentString;

/**
 * erzeugt ein MatrixAligner Objekt mit einer gegebenen Matrix
 * @param matrix die Matrix
 */
public MatrixAligner(Matrix matrix){
    this.matrix = matrix;
    if (this.matrix != null)
        this.matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der Matrix an
    this.alignerIdentString = "abstract MatrixAligner";
}

/**
 * setzt eine neue Matrix ein
 * @param m die Matrix
 */
public void setzeMatrix(Matrix m){
    if (this.matrix != null)
        this.matrix.beobachterAbmelden(this); // wir melden uns als Beobachter der alten Matrix ab
    this.matrix = m;
    if (this.matrix != null)
        this.matrix.beobachterAnmelden(this); // wir melden uns als Beobachter der neuen Matrix an
}

/**
 * Liefert den alignerIdentString. Der kann dann z.B. im GUI als Fenstertitel gesetzt werden.
 */
public String alignerInfo(){
    return alignerIdentString;
}

/**
 * Soll in Unterklassen so überschrieben werden, dass
 * ein modaler Konfigurationsdialog abgefahren wird und die Einstellungen
 * dann gestzt werden.
 * Bietet der Aligner keine Einstellmöglichkeiten, ist nichts weiter zu tun.
 * Diese Methode wird von der grafischen Oberfläche (Mavis/MACV) aufgerufen, nachdem
 * der Aligner instanttiert wurde und bevor das Alignment gestartet wurde.

```

```

*/
public void konfigurationsDialogAbfahren(){
    System.err.println("+alignerIdentString+ bietet keine Einstellmöglichkeiten.");
}

////////////////////////////////////
//
// Alignment, Observer-Behandlung
//
/** für Benachrichtigung des Observers, falls in einem Extrathread gearbeitet wird */
protected AlignmentObserver observer = null;
/** Flagge, die Abbruch des Alignmentvorganges signalisiert. Wird in der Regel in der
 * Hauptschleife der Methode abgefragt, die das Alignment berechnet */
protected boolean cancelled = false;
/** Flagge, die auf true gesetzt wird, wenn der Alignmentvorgang beendet ist.
 * Vor einem erneuten Alignen ist diese Flagge wieder auf false zu setzen */
protected boolean finished = false;
/** Flagge, die auf true gesetzt wird, wenn der Alignmentvorgang beginnt.*/
protected boolean started = false;

/**
 * ermittelt (den) Alignment-Pfad
 * (ist von den Unterklassen zu implementieren)
 */
public abstract Pfad align();

/**
 * Diese Methode startet das Alignment in einem neuen Thread.
 * Das Alignment-Ergebnis
 * wird per Observer-Mechanismus an den AlignmentObserver obs zugestellt.
 * (aus interface AlignmentObservable)
 */
public void align(AlignmentObserver obs){
    observer = obs; // Auftraggeber merken
    System.err.println(alignerIdentString+": In neuem Thread gestartet."); // Nachricht ausgeben
    Thread al = new Thread(this); // neuen Thread anlegen
    al.start(); // und starten
}

/**
 * Diese Methode setzt die Flagge zum Beenden des Alignments
 * (aus interface AlignmentObservable)
 */
public void cancel(){
    cancelled = true; // Flagge setzen
    System.err.println(alignerIdentString+": Alignmentvorgang abgebrochen"); // Nachricht ausgeben
    abmelden(); // Observer/MatrixBeobachtung v
}

/**
 * Meldet den MatrixAligner als Beobachter von der Matrix ab und vergisst
 * seinen Observer.
 * "abmelden" sollte das GUI zum MatrixAligner sagen, bevor es ihn entsorgt,
 * da die Matrix den MatrixAligner sonst noch als Beobachter registriert hat,
 * es also noch eine Referenz gibt und der Aligner folglich nicht
 * vom Garbage-Collector eingesammelt werden kann!!!
 */
public void abmelden(){
    matrix.beobachterAbmelden(this); // von der Matrix abmelden
    observer=null; // Observer vergessen
}

```

```

/**
 * run-Methode für Arbeit in einem eigenen Thread
 */
public void run(){
    align(); // Return-Wert brauchen wir nicht, der wird per ObserverInterface geliefert...
}

////////////////////////////////////
//
// Methode, die das Interface de.denkselbst.beobachtung.Beobachter verlangt
//
/**
 * Auf Änderungen in der Matrix reagieren die Aligner sehr empfindlich, nämlich
 * mit Abbruch des Alignments.
 * (aus interface Beobachter)
 */
public void beobachtetesObjektHatSichGeregt(Object o, String nachricht){
//System.err.println("s: "+started+" f: "+finished);
    if(started == true && finished == false ){ // Nur, wenn das Alignen schon begonnen wurde und noch im
        System.err.println("Beobachtetes Objekt ("+o.getClass()+") meldet Änderung:\n "+nachricht);
        System.err.println("Deshalb bricht sich "+alignerIdentString+" ab.");
        cancel();
        informObserver("Alignment abgebrochen, da die Matrix geändert wurde.",false);
    }
}

////////////////////////////////////
//
// Convenience-Methoden zur Kommunikation mit dem Observer
//
/** Zähler für informObserverZZZ */
private int notifyCount;

/**
 * Convenience-Methode: Status-Nachricht an Observer
 * (prüft auf observer != null)
 */
protected void informObserver(String text, boolean erfolg){
    if(observer!=null){
        observer.statusInformation(text,erfolg);
    }
}

/**
 * Convenience-Methode: Status-Nachricht an Observer, aber nur jede 100.
 * Nachricht wird tatsächlich weitergereicht.
 * (prüft auf observer != null)
 */
protected void informObserverZZZ(String text, boolean erfolg){
    notifyCount++;
    if(notifyCount > 100){
        if(observer!=null){
            observer.statusInformation(text,erfolg);
        }
        notifyCount = 0;
    }
}

```

```

}

/**
 * Convenience-Methode: Alignmentergebnis an Observer weitergeben
 * (prüft auf observer != null)
 * @param p der Ergebnispfad (bei Mißerfolg leeren Pfad übergeben)
 * @param erfolg wenn false, war das Alignment erfolglos und p
 * enthält nicht den Pfad (sondern ist z.B. leer, o.ä.)
 */
protected void informObserverAboutFinishedAlignment(Pfad p, boolean erfolg){
    if(observer!=null){
        observer.alignmentFertig(p, erfolg);
    }
}

}

} // Ende der Klassendefinition

```

M.5.15 de.denkselbst.matrixAligner.AlignmentObservable.java

```

package de.denkselbst.matrixAligner;

/**
 * Dieses Interface legt die Methoden fest, die ein Alignment-Objekt beherrschen
 * muss, wenn es Alignment in einem eigenen Thread anbieten will. (Observer-Muster)
 */
public interface AlignmentObservable{
    /** Starte Alignment unter Beobachtung/Einfluss von @param obs*/
    public void align(AlignmentObserver obs);
    /** AlignmentObserver kann durch Aufruf dieser Methode den Alignmentvorgang abbrechen */
    public void cancel();
}

```

M.5.16 de.denkselbst.matrixAligner.AlignmentObserver.java

```

package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Pfad;

/**
 * Es handelt sich um ein Interface, das die Methoden festlegt, die ein Objekt beherrschen
 * muss, wenn es Alignment in einem anderen Thread lostreten will und
 * über den Fortschritt benachrichtigt werden will. (Observer-Muster)
 */
public interface AlignmentObserver{
    /** diese Methode wird vom observierten MatrixAligner-Objekt
        aufgerufen, wenn der Alignmentprozess ein Ende gefunden hat. @param erfolg
        enthält einen boolschen Wert, der andeutet, ob das Unternehmen erfolgreich war, wenn
        ja, dann enthält @param p den Alignmentpfad.
    */
    public void alignmentFertig(Pfad p, boolean erfolg);
    /** Übermittelt Statusinformationen (z.B. zu Ausgabe in einer grafischen Oberfläche) */
    public void statusInformation(String s, boolean erfolg);
}

```

M.5.17 de.denkselbst.matrixAligner.AStarMatrixAligner.java

```

package de.denkselbst.matrixAligner;

```

```

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

import java.util.ArrayList;
//import java.util.Iterator;

import com.blumarsh.adt.BinomialHeap;
import com.blumarsh.adt.BinomialHeap.Node;

// Für den Konfig-Dialog
import javax.swing.JDialog;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.JCheckBox;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

/**
 * Diese Klasse ist ein MatrixAligner, der den A*-Suchalgorithmus für die Berechnung des Pfades
 * einsetzt.
 *
 * Dreht man die Suchrichtung um (rechts unten nach links oben), so kann aufgrund der
 * Methode, die die Nachfolgeknoten berechnet kein Ergebnis gefunden werden.
 */

public class AStarMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

////////////////////////////////////////////////////
//
// Konstruktor und Konfigurations-Dialog
//
/**
 * Konstruktor
 */
public AStarMatrixAligner(Matrix matrix){
    super(matrix); // Matrix durchreichen an Superklasse
    this.alignerIdentString = "AStarMatrixAligner"; // Beschreibung setzen
}

/**
 * modalen Konfigurationsdialog abfahren und Aligner entsprechend konfigurieren.
 */
public void konfigurationsDialogAbfahren(){
    ConfigDialog conf = new ConfigDialog();
}

////////////////////////////////////////////////////
//
// Flaggen, die das Verhalten des Aligners steuern
//
/** Flagge, ob Schritte nach Südwesten erlaubt sind */

```



```

protected boolean SW_allowed = false;
/** Flagge, ob Schritte nach Westen erlaubt sind */
protected boolean W_allowed = false;
/** Flagge, ob Schritte nach Nordwesten erlaubt sind */
protected boolean NW_allowed = false;
/** Flagge, ob Schritte nach Norden erlaubt sind */
protected boolean N_allowed = false;
/** Flagge, ob Schritte nach Nordosten erlaubt sind */
protected boolean NE_allowed = false;
// Schritte nach Osten, Südosten und Süden sind implizit erlaubt!

/** Wenn möglich, nur Nachfolger mit besserem Wert als Schwelle erzeugen. Findet sich
    kein solcher, sollen alle Nachfolger erzeugt werden
    use_Threshold = Flagge */
protected boolean use_Threshold = true;
/** expandThreshold = Schwelle, die die Nachfolger unterbieten müssen */
protected double expandThreshold = 0.75;

/** Ist diese Flagge gesetzt, wird der Pfad von unten rechts nach oben links gesucht, statt
    wie defaultmäßig von oben rechts nach unten links */
protected boolean reverseAlignDir = false;

//////////////////////////////////////
//
// Implementation des A*-Aligners
//
BinomialHeap open; // open - Menge von Knoten als Heap
Node[] nodes; // Tabelle aller Knoten (dient u.a. als closed)

// innere Klasse für Knoten
private class Node{
    int nodeID;
    BinomialHeap.Node heapNodeRef;
    Node cheapestAncestor;
    ArrayList successors;
    boolean open;
    boolean closed;
    double g;
    double h;
    int xpos;
    int ypos;

    Node(int id, int x, int y){
        nodeID = id;
        xpos = x;
        ypos = y;
        successors = new ArrayList(3);
        // restlichen Werte automatisch 0, bzw. null
    }

    double getF(){
        return g+h;
    }

    // Alle bekannten Nachfolger von neuen g-Kosten in Kenntnis setzen; zx, zy = Zielkoordinaten
    // Enthält möglicherweise Fehler!
    // Wird aber derzeit offenbar nicht aufgerufen...
    void update(BinomialHeap openHeap, int zx, int zy){
//System.err.println("updating");
        for(int i = 0; i < successors.size(); i++){
            int succID = ((Integer)successors.get(i)).intValue(); // ID besorgen und damit

```

```

Node succNode = nodes[succID]; // Knoten besorgen

// Hat Nachfolger schlechteren f-Wert als f'=unser g+seine Kosten+ sein h?
double gSucc = g + matrix.holeDatumVonStelle(succNode.xpos, succNode.ypos);
double hSucc = matrix.minKost(succNode.xpos, succNode.ypos, zx, zy);
double newF = gSucc + hSucc;

if( newF < succNode.getF() ){
//System.err.println("newF < succNode.getF()");
    succNode.cheapestAncestor = this; // dann auf uns verweisen
    succNode.g = gSucc; // neue g-Kosten eintragen
    succNode.h = hSucc; // neue h-Kosten eintragen
    succNode.update(openHeap, zx, zy); // Änderung propagieren

    // Wenn wir in open sind, muss der heap geändert werden!!!
    if(open){
//System.err.println("we are in open, so we change our value in the heap");
        openHeap.delete(heapNodeRef); // alten Eintrag wegwerfen
        heapNodeRef = openHeap.insert(new Integer(nodeID), getF()); // Knoten mit upgedateten f-Kos
    }
} //if besserer Wert
} //for
} //update

} //Node

/**
 * a*-Suche von (x1,y1) nach (x2,y2)
 */
private Pfad aStarSearch(int x1, int y1, int x2, int y2){

    int schleifenZaehler = 0;
    informObserver("Initialisiere",true);

    // -- Initialisierung allgemein -----
    int breite = matrix.holeBreite(); // Breite und Hoehe der Matrix
    int hoehe = matrix.holeHoehe(); // merken

    open = new BinomialHeap(); // Heap anlegen

    nodes = new Node[breite * hoehe]; // Tabelle für Knoten-Objekte anlegen
    for(int j = 0; j < hoehe; j++){ // und initialisieren
        for(int i = 0; i < breite; i++){
            int id = j*breite + i; // Knoten fortlaufend durchnummerieren
            Node node = new Node(id,i,j); // Knoten-Objekt erzeugen
            node.h = matrix.minKost(i, j, x2, y2); // h setzen
            nodes[id] = node; // und in die Tabelle eintragen
        }
    }

    // -- Initialisierung des open-Heaps -----
    int startNodeID = y1 * breite + x1; // ID des Startknoten
    Node startNode = nodes[startNodeID]; // zugehöriges Knoten-Objekt besorgen
    startNode.open = true; // Flagge setzen: Dieser Knoten ist in open
    startNode.g = matrix.holeDatumVonStelle(x1, y1); // g-Kosten setzen
    //startNode.h = matrix.minKost(x1, y1, x2, y2); // h-Kosten setzen
    startNode.heapNodeRef = open.insert(new Integer(startNodeID), startNode.getF()); // ID des Startknotens

```

```

// -----
while(true){

    schleifenZaehler++; // Lediglich ein unwichtiger Zähler, um dem Observer einen Fortschritt melden zu können
    informObserverZZZ(""+schleifenZaehler+". Iteration, "+open.size()+" Knoten sind in open.",true);

    if(open.empty()){ // Wenn open leer ist, ist die Suche gescheitert
        informObserver("Agenda leer in "+schleifenZaehler+". Iteration. Misserfolg, Abbruch.",false);
        informObserverAboutFinishedAlignment(new Pfad(), false);
        return new Pfad();
    }

    // Teste, ob cancel-Nachricht vom observer kam.
    if(cancelled) {
        informObserver("Alignment abgebrochen.",false);
        informObserverAboutFinishedAlignment(new Pfad(), false);
    }

    // besten Knoten aus open entfernen
    int bestNodeID = ((Integer)(open.removeMin().data)).intValue(); // ID des besten Knotens aus open besorgen
    Node bestNode = nodes[bestNodeID]; // Knoten per ID besorgen
    bestNode.open = false; // open-Flag auf false setzen
    bestNode.closed = true; // closed-Flag auf true setzen, Knoten ist jetzt in closed

    // Prüfen, ob bestNode der Zielknoten ist
    if(bestNode.xpos == x2 && bestNode.ypos == y2){
        Pfad pfad = new Pfad(); // Pfad-Objekt
        pfad.haengeAn(x2,y2); // Zielknoten anhängen
        Node n = bestNode.cheapestAncestor; // jeweils Vorgänger anhängen
        while(n != null){
            pfad.haengeAn(n.xpos, n.ypos);
            n = n.cheapestAncestor;
        }
        pfad.umdrehen(); // Pfad umdrehen
        informObserver("Alignment abgeschlossen in "+schleifenZaehler+". Iteration. "+open.size()+" Knoten sind in open.",true);
        informObserverAboutFinishedAlignment(pfad, true);
        return pfad; // und liefern
    }
    else{
//System.err.println("Nachfolger erzeugen");
        // Nachfolger von bestNode erzeugen
        ArrayList successors = nachfolger(bestNode.xpos, bestNode.ypos);

        // und abarbeiten
        for(int s = 0; s < successors.size(); s++){
//System.err.println(" arbeite Nachfolger "+(s+1)+"/"+successors.size()+" ab.");
            int succID = ((Integer)successors.get(s)).intValue(); // ID besorgen und damit
            Node succNode = nodes[succID]; // Knoten besorgen
            double gSucc = bestNode.g + matrix.holeDatumVonStelle(succNode.xpos, succNode.ypos); // g-Kosten

            // Prüfen, ob dieser Nachfolger schon in open bekannt ist
            if(succNode.open){
//System.err.println(" Nachf in open");
                // Wenn wir diesmal billiger zu diesem Nachfolger gekommen sind:
                if(gSucc < succNode.g){
//System.err.println(" diesmal billiger: "+gSucc+" < "+succNode.g+" bisheriges f: "+succNode.getF());
                    open.delete(succNode.heapNodeRef); // alten Eintrag wegwerfen
                }
            }
        }
    }
}

```

```

        succNode.cheapestAncestor = bestNode; // Rückverweis eintragen
        succNode.g = gSucc; // neue g-Kosten eintragen
        //succNode.h = matrix.minKost(succNode.xpos, succNode.ypos, x2, y2); // neue h-Kos
        succNode.heapNodeRef = open.insert(new Integer(succID), succNode.getF()); // Knoten mit
//System.err.println("    neues f: "+succNode.getF());
    }
}
else if(succNode.closed){ // sonst prüfen, ob dieser Nachfolger in closed enthalten ist
//System.err.println("    Nachf in closed");
    // Liste der Nachfolgeknoten von bestNode um diesen Nachfolger erweitern
    if(!((ArrayList)bestNode.successors).contains(new Integer(succNode.nodeID))) // ist dieser
        ((ArrayList)bestNode.successors).add(new Integer(succNode.nodeID));

    // Wenn wir diesmal billiger zu diesem Nachfolger gekommen sind:
    if(gSucc < succNode.g){
//System.err.println("    diesmal billiger: "+gSucc+ " < "+succNode.g+" bisheriges f: "+succNode.getF());
        succNode.cheapestAncestor = bestNode; // Rückverweis eintragen
        succNode.g = gSucc; // neue g-Kosten eintragen
        //succNode.h = matrix.minKost(succNode.xpos, succNode.ypos, x2, y2); // neue h-Kos
//System.err.println("    calling update");
        succNode.update(open, x2,y2); // Änderung propagieren dazu muss der Heap leider

    }

}
else{ // dieser Nachfolger war weder in open noch in closed
//System.err.println("    Nachf weder in open noch in closed");
    ((ArrayList)bestNode.successors).add(new Integer(succNode.nodeID)); // diesen Nachfolger i
    succNode.open = true; // dieser Nachfolger i
    succNode.g = bestNode.g + matrix.holeDatumVonStelle(succNode.xpos, succNode.ypos); //
    //succNode.h = matrix.minKost(succNode.xpos, succNode.ypos, x2, y2); // neue h-Kosten besor
    succNode.heapNodeRef = open.insert(new Integer(succNode.nodeID), succNode.getF()); // Nach

    if(succNode.cheapestAncestor == null)
/*claus?*/ succNode.cheapestAncestor = bestNode; // Rückverweis eintragen - steht nicht in claus 1
}

} // for

} // else von if-Zielknoten

}

/**
 * Innere Klasse, die für die Erzeugung der Nachfolgeknoten als
 * Datenstruktur dient
 */
private class helper{
    public int x;
    public int y;
    public double wert;
    public helper(int x, int y, double wert){
        this.x = x;
        this.y = y;
        this.wert = wert;

```

```

    }
}

/**
 * Nachfolgeknoten eines Knotens erzeugen, geliefert wird Liste von IDs
 *
 * Momentanes Problem: Wenn zusätzliche Nachfolger (NE, W, ...) erlaubt sind, ist
 * bei Verwendung der Schwelle nicht sichergestellt, dass ein Nachfolger gefunden wird,
 * der weiter in Richtung Ziel führt. Es kann aufgrund zu schlechter Bewertung zu einem
 * Zyklus SE -> NE -> SE kommen, was beim A* dazu führt, dass die Agenda sich leert und
 * kein Weg zum Ziel gefunden werden kann. Behoben werden kann das Problem, in dem
 * sichergestellt wird, dass zumindest einer der Nachfolger näher zum Ziel führt, sprich,
 * einer der drei Standartnachfolger erzeugt wird. (Später)
 */
private ArrayList nachfolger(int x, int y){
    int breite = matrix.holeBreite(); // Breite und Hoehe der Matrix
    int hoehe = matrix.holeHoehe(); // merken

    ArrayList potn = new ArrayList(3); // Potentielle Nachfolger
    double schwelle = expandThreshold; // lokale Kopie der globalen Schwelle

    // Im ersten Schritt werden alle potentiellen Nachfolgeknoten aufgesammelt.
    // (Im zweiten Schritt werden evtl. einige davon aussortiert.)
    //
    // Erzeugung der legalen "normalen" Nachfolger
    //
    // schräg rechts runter
    if(x+1 < breite && y+1 < hoehe){
        potn.add(new helper(x+1, y+1, matrix.holeDatumVonStelle(x+1, y+1) ));
    }
    // rechts
    if(x+1 < breite){
        potn.add(new helper(x+1, y, matrix.holeDatumVonStelle(x+1, y) ));
    }
    // runter
    if(y+1 < hoehe){
        potn.add(new helper(x, y+1, matrix.holeDatumVonStelle(x, y+1) ));
    }
    // sind Schritte in "ungewöhnliche" Richtungen erlaubt, werden die
    // entsprechenden Nachfolgeknoten erzeugt, falls sie legal sind.
    //
    // Westen
    if(W_allowed && x-1 > 0){
        potn.add(new helper(x-1, y, matrix.holeDatumVonStelle(x-1, y) ));
    }
    // Norden
    if(N_allowed && y-1 > 0){
        potn.add(new helper(x, y-1, matrix.holeDatumVonStelle(x, y-1) ));
    }
    // Nordosten
    if(NE_allowed && x+1 < breite && y-1 > 0){
        potn.add(new helper(x+1, y-1, matrix.holeDatumVonStelle(x+1, y-1) ));
    }
    // Südwesten
    if(SW_allowed && x-1 > 0 && y+1 < hoehe){
        potn.add(new helper(x-1, y+1, matrix.holeDatumVonStelle(x-1, y+1) ));
    }
    // NordWesten
    if(NW_allowed && (x-1 > 0) && (y-1 > 0) ){
        potn.add(new helper(x-1, y-1, matrix.holeDatumVonStelle(x-1, y-1) ));
    }
}

```

```

}

// Aus den potentiellen Nachfolgern nun die raussuchen, die es tatsächlich geschafft haben,
// also z.B. einen besseren Wert als die angegebene Schwelle haben
// Soll eine Schwelle benutzt werden, und bleibt die Nachfolgerliste nach einem
// Schleifendurchlauf leer, so wird die Schwelle nach oben korrigiert, bis sie
// >= 1.0 ist, also ausser Kraft gesetzt wurde.
ArrayList n; // Liste der Nachfolger
for(;;){
    n = new ArrayList(potn.size()); // neue Nachfolgerliste anlegen
    for(int i = 0; i < potn.size(); i++){ // potentielle Nachfolger abgrasen
        helper node = (helper)potn.get(i); // aktueller pot. Nachf.
        if(use_Threshold){ // Wenn Schwelle benutzt werden soll
            if(node.wert < schwelle){ // prüfen ob pot. Nachf. Schwelle untersch
                n.add( new Integer(node.x + node.y * breite) ); // und dann gegebenenfalls hinzufügen
            }
        }else{ // Soll keine Schwelle verwendet werden,
            n.add( new Integer(node.x + node.y * breite) ); // pot. Nachf. der Liste der Nachf. hinz
        }
    }
}
// Zumindest ein Nachfolger hat es geschafft, also Schleife verlassen
if(!n.isEmpty()) break; // Hier muss später sichergestellt werden, dass zumindest
// ein Nachfolger enthalten ist, der weiter zum Ziel
// führt, o.ä., so dass kein Zyklus entsteht, der die
// Agenda leer werden lässt.

// Es soll gar keine Schwelle benutzt werden, also Schleife verlassen
if(!use_Threshold) break;

// Ist die Schwelle >= 1.00, dann Schleife verlassen
if(schwelle >= 1.00) break;

// Ansonsten gilt: Schwelle soll benutzt werden, aber Nachfolgerliste ist leer,
// also muss die Schwelle korrigiert werden.
schwelle = 1.05; // Vielleicht optional eine stufenweise Erhöhung um jeweils die Hälfte der aktuell

//System.err.println("Schwelle: "+schwelle); // aktuellen lokalen Schwellwert ausgeben
}

if(n.isEmpty()) System.err.println("Konnte keine Nachfolger zu "+x+", "+y+" erzeugen.");
//else System.err.println("."); // Alive-Zeichen ausgeben

return n;
}

/**
 * ermittelt (den) Alignment-Pfad
 */
public Pfad align(){
    finished = false;
    cancelled = false;

```

```

started = true;

Pfad p;
if(reverseAlignDir){
    p = aStarSearch(matrix.holeBreite()-1,matrix.holeHoehe()-1,0,0);
}else p = aStarSearch(0,0,matrix.holeBreite()-1,matrix.holeHoehe()-1);

finished = true;
started = false;
return p;
}

////////////////////////////////////
//
// Innere Klasse, die innerhalb einer grafischen Oberfläche
// einen Konfigurationsdialog darstellt und die vom Benutzer
// eingestellten Werte in den Aligner einträgt.
//
/**
 * Innere Klasse, die innerhalb einer grafischen Oberfläche
 * einen Konfigurationsdialog darstellt und die vom Benutzer
 * eingestellten Werte in den Aligner einträgt.
 */
protected class ConfigDialog extends JDialog implements ActionListener{

    JCheckBox[] additionalDirections = new JCheckBox[5];
    JButton okButton;
    JCheckBox useThresholdCheckBox, alignDirCheckBox;
    JTextField ThresholdTextField;

    /** Konstruktor. Legt den modalen Dialog an */
    public ConfigDialog(){
        // Owner, Title, Modal
        super((java.awt.Frame)null,"A* - Einstellungen",true);

        //JPanel panel = new JPanel(new GridLayout(5,2));
        JPanel panel = new JPanel();
        String[] texts = {"SW","W","NW","N","NE"};
        for(int i = 0; i < 5; i++){
            additionalDirections[i] = new JCheckBox(texts[i]);
            panel.add(additionalDirections[i]);
        }

        JPanel panell = new JPanel(new GridLayout(2,0));
        panell.add(new JLabel("Zusätzlich zu W, SW, S erlaubte Richtungen"));
        panell.add(panel);

        useThresholdCheckBox = new JCheckBox();
        ThresholdTextField = new JTextField();
        ThresholdTextField.setText("0.75");

        JPanel panel2 = new JPanel(new GridLayout(2,2));
        panel2.add(new JLabel("Benutze Schwelle"));
        panel2.add(useThresholdCheckBox);
        panel2.add(new JLabel("Schwellwert für Nachfolger"));
        panel2.add(ThresholdTextField);

```

```

JPanel panel3 = new JPanel(new GridLayout(0,2));
panel3.add(new JLabel("von rechts unten nach links oben alignen"));
alignDirCheckBox=new JCheckBox();
panel3.add(alignDirCheckBox);

okButton = new JButton("OK");
okButton.addActionListener(this);

getContentPane().setLayout(new GridLayout(3,0)); // 4,0, wenn panel3 auch Platz haben soll

getContentPane().add(panel1);
getContentPane().add(panel2);
//getContentPane().add(panel3); // Umgekehrte Alignmentrichtungs-Option nicht anzeigen, da kaputt

getContentPane().add(okButton);

pack();
show();
}

public void actionPerformed(ActionEvent e){

// Aha, ok wurde gedrückt...
if(e.getSource() == okButton){

// Flaggen für erlaubte Wege setzen
SW_allowed = ((JCheckBox)additionalDirections[0]).isSelected();
W_allowed = ((JCheckBox)additionalDirections[1]).isSelected();
NW_allowed = ((JCheckBox)additionalDirections[2]).isSelected();
N_allowed = ((JCheckBox)additionalDirections[3]).isSelected();
NE_allowed = ((JCheckBox)additionalDirections[4]).isSelected();

//reverseAlignDir = alignDirCheckBox.isSelected(); // Funktioniert derzeit nicht richtig, Nachfol

use_Threshold = useThresholdCheckBox.isSelected();
// Wenn die Schwelle denn benutzt werden soll, den entsprechenden Wert besorgen
if(use_Threshold){
    expandThreshold = Double.parseDouble(ThresholdTextField.getText());
    if(expandThreshold < 0 || expandThreshold > 1.0) expandThreshold = 0.75; // Im Zweifelsfall
    // Das Textfeld entweder zu Typchecking verdonnern oder durch einen Slider ersetzen...
}

// Einstellungen ausgeben
System.err.println("A* - Einstellungen:");
System.err.println("SW_allowed = "+ SW_allowed);
System.err.println("W_allowed = "+ W_allowed);
System.err.println("NW_allowed = "+ NW_allowed);
System.err.println("N_allowed = "+ N_allowed);
System.err.println("NE_allowed = "+ NE_allowed);
System.err.println("");
System.err.println("use_Threshold = "+ use_Threshold);
System.err.println("expandThreshold = "+ expandThreshold);
System.err.println("reverseAlignDir = "+ reverseAlignDir);

hide();
dispose();
}

```



```

    }
}

////////////////////////////////////
//
// Main-Methode
//
/** winziger Test: Lese Matrix aus Datei args[0] ein und aligne mal*/
public static void main (String[] args){
    Matrix m = new Matrix(args[0]);
    AStarMatrixAligner asma = new AStarMatrixAligner(m);

    Pfad p = asma.aStarSearch(0,0,m.holeBreite()-1,m.holeHoehe()-1);
    System.out.println(""+p);
}
}
}

```

M.5.18 de.denkselbst.matrixAligner.BestCellMatrixAligner.java

```

package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

```

```
/**
```

Diese Klasse löst das folgende Problem:

Im de-news Korpus tritt das Phänomen auf, dass die Reihenfolge der Nachrichtenblöcke in der englischen Datei nicht mit der deutschen Datei übereinstimmt, obwohl zu jedem Block eine Übersetzung vorhanden ist.

Idee aus dem Projekt / Arno: Man nehme aus beiden Dateien nur die Überschriften und aligne diese mit einem simplen Verfahren, das Überkreuzungen zulässt. Auf diese Weise wird ein Mapping von englischen zu deutschen Überschriften erzeugt. Anhand dieses Mappings lassen sich die Dateien in dieselbe Reihenfolge bringen.

Dieser MatrixAligner:

- a) sucht die beste Matrixzelle,
- b) trägt ihre Koordinaten als Mapping ein,
- c) zerstört die gesamte Spalte und Zeile, in der die Zelle liegt
- und d) wiederholt diese Schritte, bis zu jeder Koordinate der kürzeren Matrixdimension eine entsprechende Koordinate gefunden wurde, also zu jeder Zeile (Spalte) eine Spalte (Zeile) zugeordnet wurde.

Auf diese Weise wird ein 1:1 Mapping von X zu Y (also englischer zu deutscher Überschrift) erzeugt. Momentan wird das Ergebnis als Pfad geliefert. Das Ergebnis ist korrekt, führt aber zu einer unschönen Darstellung in Mavis, da die Pfadkoordinaten nicht in geeigneter Reihenfolge im Pfad stehen, um in einem Linienzug gezeichnet zu werden.

Konvertiert man den Pfad in eine Kokszuordnung, ist die Darstellung ok, die Reihenfolge jedoch unverändert.

Ist die Matrix nicht quadratisch, gehen einige Spalten(Zeilen) leer aus, da auf einer 1:1

```

    Zuordnung bestanden wird.
    */

public class BestCellMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

    /** Konstruktor */
    public BestCellMatrixAligner(Matrix matrix){
        super(matrix);
        this.alignerIdentString = "BestCell-1:1-MatrixAligner";    // Beschreibung setzen
    }

    /** Alignment-Methode */
    public Pfad align(){

        // Gib Statusinformation aus. (Wird an den Observer geschickt, der möglicherweise für eine Ausgabe sorgt)
        informObserver("Beginne 1:1 bestCell - Alignment", true);

        // Ergebnispfad
        Pfad p = new Pfad();    // Hier muss eigentlich eine KoksZuordnung hin!
                                // Das ist zu ändern!!!!

        // Höhe und Breite der Matrix besorgen
        int x = matrix.holeBreite();
        int y = matrix.holeHoehe();

        // lokale Kopie der Matrix erstellen, die bedenkenlos verändert werden kann
        Matrix lokal = new Matrix(matrix.kopieDerDaten());

        // Wert und Koordinaten der besten Zelle
        double bestValue;
        int bestX;
        int bestY;

        // Welche Dimension der Matrix ist kleiner?
        int mindim;

        boolean useX;
        if(x < y){ useX = true; mindim = x; }
        else{ useX = false; mindim = y; }

        // merken, ob Zeile schon für eine Zuordnung "benutzt" wurde
        boolean[] xBelegt = new boolean[mindim];    // implizit mit false initialisiert

        // Schleifenende-Flagge
        boolean fertig = false;

        // solange wir noch nicht fertig sind
        while(!fertig){

            // Teste, ob cancel-Nachricht vom observer kam.
            if(cancelled) {
                informObserver("Alignment abgebrochen.",false);
                informObserverAboutFinishedAlignment(new Pfad(), false);
                return new Pfad();
            }

            // Gib Statusinformation aus.
            informObserver("Suche beste Zelle.", true);

```

```

// besorge Koordinaten der besten Zelle
bestX = lokal.getMinX();
bestY = lokal.getMinY();

// Nimm gefundene Koordinate in Lösungspfad auf
p.haengeAn(bestX, bestY);

// Gib Statusinformation aus.
informObserver("Habe beste Zelle gefunden (" + bestX + ", " + bestY + ").", true);

// merke, dass diese Spalte versorgt wurde
if(useX) xBelegt[bestX] = true;
else xBelegt[bestY] = true;

// Gib Statusinformation aus.
informObserver("Lösche zugehörige Zeile und Spalte.", true);

// setze zur besten Zelle gehörige Zeile und Spalte auf unerreichbar schlechte Werte
lokal.fillRow(bestY, 666.6); // ein- / ausschaltbar machen!
lokal.fillColumn(bestX, 666.6);

// Gib Statusinformation aus.
informObserver("Prüfe, ob fertig.", true);

// Prüfen, ob alle Spalten versorgt sind
boolean test = true;
for(int i = 0; i < xBelegt.length; i++){
    test = test & xBelegt[i]; // Wenn eine Spalte nicht versorgt ist (=false), dann wird
}
fertig = test;
}

// Gib Statusinformation aus und liefere den Pfad.
informObserver("BestCell1:1-Alignment abgeschlossen", true);
informObserverAboutFinishedAlignment(p, true);
return p;
}

/** Die run-Methode ruft lediglich align() auf. Ergebnisse werden über das Observer-Modell
    an den Interessenten geliefert. (Daher wird der Rückgabewert der Methode an dieser
    Stelle ignoriert.)
    */
public void run(){
    align();
}

/** winziger Test: Lese Matrix aus Datei args[0] ein und gebe sie auf System.out aus. */
public static void main (String[] args){
    Matrix m = new Matrix(args[0]);
    MatrixAligner aligner = new BestCellMatrixAligner(m);
    System.out.println("Alignment: " + aligner.align());
}
}

```

M.5.19 de.denkselbst.matrixAligner.LMEDMatrixAligner.java

```

package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

import java.io.*;
import java.util.StringTokenizer;
import java.util.Iterator;

/**
 * Dieser Aligner basiert auf dem Levenshtein-Minimum-Edit-Distance Algorithmus. Er bietet eine bessere Laufzeit
 * als der AStarMatrixAligner, aber liefert auch schlechtere Ergebnisse.
 * Er entstand recht früh und in Zusammenarbeit mit Philip Reuter.
 */
public class LMEDMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

    double[][] leven;    // Levenshteinmatrix
    double[][] werte;    // Abstandsmatrix
    int levenX, levenY;  // Dimensionen

    /** Konstruktor */
    public LMEDMatrixAligner(Matrix matrix){
        super(matrix);
        this.alignerIdentString = "LMEDMatrixAligner";    // Beschreibung setzen
        levenX = matrix.holeBreite();
        levenY = matrix.holeHoehe();
        leven = new double[levenX][levenY];
        werte = new double[levenX][levenY];
        initWerte();
        initLeven();
    }

    private void initWerte(){
        double min = cost(0,0);
        double max = cost(0,0);
        double eps = 0.1;
        // Werte einlesen, dabei max und min berechnen
        for(int i=0; i < levenX; i++){
            for(int j=0; j < levenY; j++){
                werte[i][j] = cost(i,j);
                if(werte[i][j] < min){
                    min = werte[i][j];
                }
                if(werte[i][j] > max){
                    max = werte[i][j];
                }
            }
        }
        // Werte spreizen: (((x-min)*(1-eps))/(max-min))+eps
        for(int i=0; i < levenX; i++){
            for(int j=0; j < levenY; j++){
                werte[i][j] = (((werte[i][j]-min)*(1-eps))/(max-min))+eps;
            }
        }
    }

    private void initLeven(){
        // Extra-Spalte berechnen
        leven[0][0] = werte[0][0];
    }

```

```

for(int j=1; j < levenY; j++){
    leven[0][j] = leven[0][j-1] + werte[0][j];
}
// Extrazeile und Rest berechnen
for(int i=1; i < levenX; i++){
    leven[i][0] = leven[i-1][0] + werte[i][0];

    for(int j=1; j < levenY; j++){
        leven[i][j] = minimalerVorgaenger(i,j) + werte[i][j];
    }
}
}

private double cost (int x, int y){
    return matrix.holeDatumVonStelle(x,y);
}

// die geringsten Kosten zu x1,y1
private double minimalerVorgaenger(int x, int y){
    double diagonal = leven[x-1][y-1];
    double horizontal = leven[x-1][y];
    double vertikal = leven[x][y-1];
    return min(min(diagonal, horizontal), vertikal);
}

private double min(double c1, double c2){
    if (c1 <= c2) return c1;
    return c2;
}

public double get(int xx, int yy){
    return matrix.holeDatumVonStelle(xx,yy);
}

public int getXDim(){
    return matrix.holeBreite();
}

public int getYDim(){
    return matrix.holeHoehe();
}

/** Align-Methode */
public Pfad align(){

    finished = false;
    started = true;
    cancelled = false;
    int zzz = 0;          // Alive-Counter

    informObserver("LMED sprintet los.",true);    // Convenience-Methode aus der Oberklasse

    Pfad p = new Pfad();

    int i = matrix.holeBreite()-1;
    int j = matrix.holeHoehe()-1;

    p.haengeVorneAn(i,j);

```

```

while( i>0 && j>0){

    zzz++;
    informObserver("LMED denkt angestrengt nach... "+zzz,true);

    double diagonal    = leven[i-1][j-1];
    double horizontal  = leven[i-1][j];
    double vertikal    = leven[i][j-1];
    //double minV = min(min(diagonal,horizontal),vertikal);
    //double minV = minimalerVorgaenger(i,j);

    if(leven[i][j] - werte[i][j] == diagonal){
        i--;
        j--;
    }
    else if(leven[i][j] - werte[i][j] == horizontal){
        i--;
    }else {
        j--;
    }
    p.haengeVorneAn(i,j);

}

while(i > 0){
    p.haengeVorneAn(i-1,0);
    i--;

    zzz++;
    informObserver("LMED denkt angestrengt nach... "+zzz,true);
    if(cancelled) {
        informObserver("LMED Alignment abgebrochen.",false);
        informObserverAboutFinishedAlignment(new Pfad(), false);
    }

}

while(j > 0){
    p.haengeVorneAn(0,j-1);
    j--;

    zzz++;
    informObserver("LMED denkt angestrengt nach... "+zzz,true);
    if(cancelled) {
        informObserver("LMED Alignment abgebrochen.",false);
        informObserverAboutFinishedAlignment(new Pfad(), false);
    }

}

informObserver("LMED Alignment abgeschlossen.",true);
informObserverAboutFinishedAlignment(p, true);
finished = true;
started = false;
//System.out.println(p);
return p;
}
}

```

M.5.20 de.denkselbst.matrixAligner.BFSMatrixAligner.java

```

package de.denkselbst.matrixAligner;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.Comparator;
import java.util.Arrays;

import com.bluemarsh.adt.BinomialHeap;

/**
 * Dieser MatrixAligner verwendet eine Best-First-Suchstrategie. Die hier vorliegende Implementierung
 * diene vor allem dem Herumexperimentieren mit Strafen und eignet sich nur für kleine Matrizen. Der AStarMatr
 * ist schneller und benötigt weniger Speicher, liefert aber im Standardfall das gleiche Ergebnis.
 */
public class BFSMatrixAligner extends MatrixAligner implements AlignmentObservable, Runnable{

    private int x,y;
    private float steigung;

    // Fürs bfs-alignen: maximal ist maxHoriz:maxVert als Zuordnung in einem Block möglich!
    // Wege, die ein längeres vertikale, horizontales, rechts-hoch diagonales oder linksrunterdiagonales
    // Stück enthalten, als die folgenden Schwellen angeben, werden gelöscht
    private int maxHoriz = 2; // 2
    private int maxVert = 2; // 2

    private int maxHoRe = 2; // 2 // rechts hoch
    private int maxLiRu = 2; // 2 // links runter
    private boolean strangeDirsAllowed = false; //false (zu teuer!)// die beiden obigen Moves erlaubt?

    // Wie weit darf der Pfad von der Diagonalen entfernt sein?
    private int maxDistDiagX = 2; //3
    private int maxDistDiagY = 2; //3

    // Schwelle, die das Distanzmass unterschreiten muss, wenn Bewegung nach
    // hoch rechts (hore) oder liru (links runter)
    private double horeSchwelle = 0.3;
    private double liruSchwelle = 0.3;

    // Strafen für nichtdiagonale Bewegung
    private boolean bestrafeNichtdiagonaleBewegung = false; //false;
    private double rechtsStrafe = 0.1;
    private double runterStrafe = 0.1;

    // Wieviel kostet ein Schritt (Kante)
    private double kantenkosten = 0.2;
    private boolean benutzeKantenkosten = false; //true;
    // Gesamtkosten = Knotenkosten + Kantenkosten?

    public BFSMatrixAligner(Matrix matrix){

```

```

    super(matrix);
    this.alignerIdentString = "BFSMatrixAligner"; // Beschreibung setzen
    x = this.matrix.holeBreite();
    y = this.matrix.holeHoehe();

    /*
     // nur für TestZwecke
     if (x > 16) x = 16;
     if (y > 12) y = 12;
    */

    float xx = (float) x;
    float yy = (float) y;
    steigung = yy / xx;

}

// Kosten eines Knoten liefern
public double kostenVonKnoten(Knoten k){
    return matrix.holeDatumVonStelle(k.xpos,k.ypos);
}

// Nachfolger eines Knoten liefern
public ArrayList nachfolger(Knoten k){
    ArrayList n = new ArrayList();
    int x = k.xpos;
    int y = k.ypos;
    // Wo liegen die entsprechenden Diagonalen?
    int diaX = (int) (y / steigung);
    int diaY = (int) (x * steigung);
    int distX = 0;
    int distY = 0;

    if(x>0) distX = Math.abs(x - diaX);
    if(y>0) distY = Math.abs(y - diaY);

    // Prüfen, ob zu weit von der Diagonalen entfernen
    if( steigung < 1.0){
        if(distX > maxDistDiagX){
            //System.out.println("x zu weit weg");
            return new ArrayList();
        }
    }else{
        if(distY > maxDistDiagY){
            //System.out.println("y zu weit weg");
            return new ArrayList();
        }
    }

    /*
    if (!(distY <= maxDistDiagY && distX <= maxDistDiagX)){
        //System.out.println("Zu weit weg! Steigung:"+steigung+" (maxx:"+maxDistDiagX+" x:"+distX+" ) ; maxy:"
        return new ArrayList();
    }
    */
}

```



```

// schräg rechts runter
if(x+1 < matrix.holeBreite() && y+1 < matrix.holeHoehe()){
    n.add(new Knoten(x+1, y+1));
}

// Alles, was nicht diagonal läuft, wird leicht bestraft
// Alternative: Pauschalstrafe für alle = Luftlinie zum Zielpunkt

// rechts
if(x+1 < matrix.holeBreite()){
    Knoten knot=new Knoten(x+1, y);
    if(bestrafeNichtdiagonaleBewegung){knot.gibStrafe(rechtsStrafe);}
    n.add(knot);
}

// runter
if(y+1 < matrix.holeHoehe()){
    Knoten knot=new Knoten(x, y+1);
    if(bestrafeNichtdiagonaleBewegung){knot.gibStrafe(runterStrafe);}
    n.add(knot);
}

// Optional: Bewegung in "seltsame" Richtungen
if(strangeDirsAllowed){
    // schräg rechts hoch (evtl löschen!)
    if(x+1 < matrix.holeBreite() && y-1 >= 0 && y-1 < matrix.holeHoehe() && matrix.holeDatumVonStelle(x+1,y-1) > 0){
        n.add(new Knoten(x+1, y-1));
    }

    // schräg links runter (evtl. löschen)
    if(x-1 >= 0 && x-1 < matrix.holeBreite() && y+1 > 0 && y+1 < matrix.holeHoehe() && matrix.holeDatumVonStelle(x-1,y+1) > 0){
        n.add(new Knoten(x-1, y+1));
    }
}
// endif strangedirs:
}

// Mal sehen, aus Spaß diese(n) Nachfolger mitanhängen...
//if(y-1 > 0) n.add(new Knoten(x, y-1));
//if(x-1 > 0) n.add(new Knoten(x-1, y));

//Dann aber auch Zyklencheck nötig! Haben wir...
/**/

return n;
}

// innere Klasse Knoten
public class Knoten{
    int xpos;
    int ypos;
    double kosten;

    public Knoten(int x, int y){
        xpos = x;
        ypos = y;
        kosten = kostenVonKnoten(this);
    }
}

```

```

    public double kosten(){
        return kosten;
    }

    public void gibStrafe(double strafe){
        kosten += strafe;
    }
}

public class Weg{
    ArrayList weg;
    double kosten;

    int vMov; //vMovementSinceNumberOfSquares;
    int hMov; //hMovementSinceNumberOfSquares;
    int hore; // rechts hoch count
    int liru; // links runter count
    int lastX; // To track Movement
    int lastY; // ""

    public Weg(){
        weg = new ArrayList();
        kosten = 0;
        vMov=0; // Noch nicht bewegt.
        hMov=0; // Noch nicht bewegt.
        lastX = lastY = 0; // ""
    }

    public void haengeKnotenAn(Knoten k){
        // Laufrichtung ermitteln
        int x = k.xpos;
        int y = k.ypos;

        if(x-lastX != 0 && y-lastY == 0){ // Horizontalbewegung (nur X-Achse)
            hMov++;
            vMov=0;
            hore=0;
            liru=0;

            //System.out.println("hMov++    "+hMov);
        }
        else if(y-lastY != 0 && x-lastX == 0){ // Vertikalbewegung (nur Y-Achse)
            vMov++;
            hMov=0;
            hore=0;
            liru=0;
            //System.out.println("hMov++    "+vMov);
        }
        else if(x-lastX>0 && y-lastY<0){ // rechts hoch
            vMov=0; // nur horiz. u. nur vert. Bewegung also seit 0 Knoten
            hMov=0;
            hore++;
            liru=0;
        }
        else if(x-lastX<0 && y-lastY>0){ // links runter
            vMov=0; // nur horiz. u. nur vert. Bewegung also seit 0 Knoten
            hMov=0;
            hore=0;
            liru++;
        }
        else{ // Bewegung auf beiden Achsen

```

```

        vMov=0; // nur horiz. u. nur vert. Bewegung also seit 0 Knoten
        hMov=0;
        hore=0;
        liru=0;
    }
    lastX = x;
    lastY = y;

    weg.add(k);
    kosten += k.kosten();
    //System.out.println("haenge an, Kosten gesamt: "+kosten);
}

public boolean bistDuZuLangeHorizontalOderVertikalGelaufen(){
    //if(hMov>=maxHoriz || vMov>=maxVert) System.out.println("maximum vert/horiz violated.");
    //System.out.println("weg hMov:"+hMov+" vMov:"+vMov+" violatedLimits:"+(hMov<maxHoriz && vMov<maxVert));
    return (!(hMov<maxHoriz && vMov<maxVert && hore < maxHoRe && liru < maxLiRu));
}

public Iterator iterator(){
    return weg.iterator();
}

public double kosten(){
    /* double sum = 0;
    for (int i = 0; i < weg.size(); i++){
        sum += ((Knoten)weg.get(i)).kosten(); // Kosten der Knoten aufsummieren
        // Hier später noch die Kosten der Kante hinzuaddieren!!!
    }
    return sum;*/

    // Weit weg vom Ziel sein wird bestraft, allerdings unterschätzend
    double entfernungsPauschale = 0;

    int entfSchritte = 0;

    double kost=0;

    // Entfernungspauschale Strafe besorgen
    if(!weg.isEmpty()){
        Knoten k = (Knoten) weg.get(weg.size()-1);
        entfernungsPauschale = matrix.wasSindVonHierDieUnterschaetztenKostenZurRechtenUnterenEcke(k.xpos, k.ypos, k);
        entfSchritte = matrix.wievieleSchritteVonHierZurRechtenUnterenEcke(k.xpos, k.ypos);
        //System.out.println("Entfernungspauschale von ("+k.xpos+", "+k.ypos+") zum Ziel: "+entfernungsPauschale);
        //System.out.println("Gesamtkosten: "+kosten + entfernungsPauschale);
    }

    kost = kosten + entfernungsPauschale;

    kost = kost * kost;

    // Wenn Kanten als Kosten auch mitgezählt werden sollen:
    if (benutzeKantenkosten){System.out.println("benutzeKantenkosten");
        kost += entfSchritte * kantenkosten;
        // kost += (weg.size()-1) * kantenkosten;
    }
    //System.out.println("Kosten: "+kost);
    return kost;
}

```

```

public boolean enthaeltstDuDiesenKnoten(Knoten k){
    if (weg.contains(k)) System.out.println("potentieller Zyklus");
    return weg.contains(k);
}

public boolean endestDuInDiesemKnoten(Knoten k){
    return endestDuIn(k.xpos, k.ypos);
}

public boolean endestDuIn(int x, int y){
    if(weg.isEmpty()) return false;
    int l = weg.size() - 1;
    Knoten k = (Knoten)weg.get(l);
    if(x == k.xpos && y == k.ypos) return true;
    return false;
}

public ArrayList nachfolgeWege(){
    if(weg.isEmpty()) return new ArrayList();
    int l = weg.size() - 1;
    Knoten letzter = (Knoten)weg.get(l);
    ArrayList n = nachfolger(letzter);

    ArrayList nwege = new ArrayList();
    for(int i = 0; i < n.size(); i++){
        // Zyklencheck
        if(enthaeltstDuDiesenKnoten((Knoten)n.get(i))) continue;
        // Weg kopieren
        Weg neuerWeg = new Weg();
        for(int j = 0; j < weg.size(); j++){
            neuerWeg.haengeKnotenAn((Knoten)weg.get(j));
        }
        // Nachfolgeknoten anhängen
        neuerWeg.haengeKnotenAn((Knoten)n.get(i));
        //System.err.println("neuerWegKosten: "+neuerWeg.kosten()+" alter: "+kosten());
        // Prüfen, ob maximale horizontale oder vertikale Bewegung am Stück überschritten
        if(neuerWeg.bistDuZuLangeHorizontalOderVertikalGelaufen()){
            //System.out.println("foo, zu lange in x oder y gelaufen, weg damit!!");
            continue;
        }
        // Alles klar, Weg ist gültig, also in die Nachfolgerliste aufnehmen!
        nwege.add(neuerWeg);
    }
    //System.out.println("+"+nwege.size());
    return nwege;
}

// Billig Textausgabe
public void print(){
    //System.out.println("...");
    for(int i = 0; i < weg.size(); i++){
        System.out.print("[ "+((Knoten)weg.get(i)).xpos+", ");
        System.out.print("((Knoten)weg.get(i)).ypos+" ]");
    }
    //System.out.println("...");
}

}

/** Innere Klasse: Agenda für die Breitensuche */
public static class Agenda{

```

```

BinomialHeap heap;

public Agenda(){
    heap = new BinomialHeap();
}

public boolean bistDuLeer(){
    return heap.empty();
}

public int size(){
    return heap.size();
}

    // Einen Weg einfügen
public void fuegeEinenWegEin(Weg w){
    add(w.kosten(),w);
}

    // Liste von Wegen einfügen
public void fuegeWegeEin(ArrayList wege){
    for(int i = 0; i < wege.size(); i++){
        fuegeEinenWegEin((Weg) wege.get(i));
    }
}

/** Objekt einfügen */
public void add(double val, Object o){
    heap.insert(o,val);
}

public Weg billigster(){
    return (Weg) heap.removeMin().data;
}

public void behalteNurSoviele(int anzahl){

    BinomialHeap h2 = new BinomialHeap();
    BinomialHeap.Node n;

    for(int i=0; i< anzahl; i++){
        n=heap.removeMin();
        h2.insert( n.data, ((Weg)n.data).kosten());
    }
    heap = h2;
}

}

// Mittels BFS Pfad von 0,0 nach x,y suchen
public Weg bfsalign(){

    cancelled=false;

```

```

// Agenda initialisieren
Weg start = new Weg();
start.haengeKnotenAn(new Knoten(0,0));
Knoten ziel = new Knoten(matrix.holeBreite()-1, matrix.holeHoehe()-1);
Agenda ag = new Agenda();
ag.fuegeEinenWegEin(start);

for(;;){
//System.out.println("Es sind "+ag.size()+" Wege in der Agenda.");

// observer Statusbericht geben
if(ag.size()==0) informObserver("Die Agenda ist leer.", false);
else informObserverZZZ("Es sind "+ag.size()+" Wege in der BFS-Agenda.",true);

if(cancelled) {
    informObserver("Alignment abgebrochen.",false);
    informObserverAboutFinishedAlignment(new Pfad(), false);
    observer=null; // Observer vergessen
}

// Agenda leer? -> misserfolg.
if(ag.bistDuLeer()){
    return new Weg();
}

// cancelled Flagge gesetzt?
if(cancelled){
//System.err.println("abgebrochen!");
    informObserverAboutFinishedAlignment(new Pfad(), false);
    observer=null; // Observer vergessen
///// Irgendwo muss noch geregelt werden, dass sich diese Aligner bei misserfolg automatisch
///// von der Matrix abmeldet!!!
    return new Weg();
}

// ersten Weg rausholen
Weg billig = ag.billigster();

// am Ziel? -> Weg liefern
if(billig.endestDuInDiesemKnoten(ziel)){
    System.err.println("Ziel erreicht: ");
// System.err.println("Der Wehk kostät: "+billig.kosten()+"! violation:"+billig.bistDuZuLangeHorizon
    billig.print();
    // Observer benachrichtigen.
    informObserver("Alignment abgeschlossen. "+ag.size()+" Wege.",true);
    // Observer den Weg als Pfad geben

    Iterator it = billig.iterator();
    Pfad pf = new Pfad();
    Knoten kn=null;
    while(it.hasNext()){
        // Weg in Pfad umwandeln
        kn = (Knoten)it.next();
        pf.haengeAn(kn.xpos,kn.ypos);
    }
    informObserverAboutFinishedAlignment(pf, true); // und beim Observer abliefern...
}

```

```

        observer=null; // Observer vergessen
        //von der Matrix abmelden?!?!
        return billig;
    }
    else{ // Nein, expandiere und fuege Nachfolger ein
        ag.fuegeWegeEin(billig.nachfolgeWege());
    }
}
}

/** Alignment als Pfad liefern */
public Pfad align(){

    finished = false;
    cancelled = false;
    started = true;

    Iterator it = bfsalign().iterator(); // bfsalign liefert Weg
    Pfad p = new Pfad();
    Knoten k=null;

    while(it.hasNext()){ // Weg in Pfad umwandeln
        k = (Knoten)it.next();
        p.haengeAn(k.xpos,k.ypos);
    }

    finished = true;
    started = false;
    return p; // Pfad liefern
}

/** run-Methode ruft nur bfsalign() auf... */
public void run(){
    bfsalign();
}

/** Test / Demo: Lese Matrix aus Datei args[0] ein und gebe Alignment auf System.out aus. */
public static void main (String[] args){
    Matrix m = new Matrix(args[0]);
    MatrixAligner bfs = new BFSMatrixAligner(m);
    System.out.println("Alignment: "+bfs.align());
}
}

```

M.5.21 de.denkselbst.kommandozeile.Align.java

```

package de.denkselbst.kommandozeile;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrixAligner.AStarMatrixAligner;

import java.io.IOException;
import java.io.BufferedReader;
import java.io.InputStreamReader;

```

```

/**
 * Die Main-Methode dieser Klasse liest Matrizen von STDIN und
 * schreibt Alignments nach STDOUT.
 * Die wichtigsten Schritte sind es, den Eingabestrom zu besorgen,
 * ein Matrixobjekt und ein damit verbundenes MatrixAlignerobjekt zu
 * erstellen, die Matrix aus STDIN zu füllen, zu normalisieren,
 * den MatrixAligner das Alignment (als Pfad) finden lassen, den
 * Pfad in eine KoksZuordnung umzuwandeln und das
 * Ergebnis auf STDOUT auszugeben.
 *
 * Da es offensichtliche Performanzvorteile gibt, wenn nicht für
 * jede Matrix einmal die JAVA-VM gestartet werden muss, können
 * align mehrere Aufträge pro Aufruf erteilt werden. Dazu muss der
 * erste Parameter ein Integer-Wert sein, der die Anzahl der Aufträge
 * enthält. Um den obigen Code-Ausschnitt herum existiert eine Schleife.
 * Wird align ohne Angabe der Auftragsanzahl aufgerufen, wird diese
 * Schleife nur einmal durchlaufen.
 */
public class Align{
    public static void main(String args[]) throws IOException{
        // Stdin-Eingabestrom besorgen.
        BufferedReader puff = new BufferedReader(new InputStreamReader(System.in));

        /**
         * Anmerkung: die Methode matrix.setzeDatenAusStdin() kann in diesem Szenario nicht
         * verwendet werden, da der BufferedReader des ersten Matrix-Objektes seinen Puffer
         * mit Daten aus Stdin auffüllt, die schon zur zweiten Matrix gehören. Diese gepufferten
         * Daten gehen verloren, sobald das erste Matrix-Objekt zerstört wird. Damit wird der
         * Ansatz unbrauchbar. Um dies zu verhindern, wird der BufferedReader von Main gehalten,
         * und die Matrix-Objekte lesen nur so viel, wie sie benötigen.
         */

        int anzahl = 0;                // Anzahl zu lesender Matrizen

        if(args.length == 0){
            anzahl = 1;                // eine Matrix ist zu lesen
        }
        else if(args.length == 1){
            anzahl = Integer.parseInt(args[0]); // 1. Parameter enthält Matrixanzahl
        }else{
            System.err.println("Kommandozeilenaligner liest Matrix von stdin, schreibt alignment nach stdout");
            System.exit(1);
        }

        // Aufträge abarbeiten
        while(anzahl-->0){

            // Matrixobjekt anlegen
            Matrix matrix = new Matrix();

            // Alignerobjekt anlegen
            AStarMatrixAligner aligner = new AStarMatrixAligner(matrix);

            // besorge Matrix aus Stdin, Aligner bekommt die Matrixänderung
            // automatisch von der Matrix mitgeteilt
            matrix.setzeDatenAusGepuffertemLeser(puff);

            // Normalisiere Matrix
            matrix.normalisiereDich();

```



```

// aligne
Pfad p = aligner.align();

// schreibe Ergebnis
System.out.print(p.toWagnerString(matrix.holeQuelle(),matrix.holeInfo()));
}
}
}

```

M.5.22 de.denkselbst.kommandozeile.HeadingMapper4Arno.java

```

package de.denkselbst.kommandozeile;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrixAligner.BestCellMatrixAligner;
import de.denkselbst.matrixAligner.MatrixAligner;

import java.io.IOException;

/**
 * Die Main-Methode dieser Klasse liest Matrizen von Stdin
 * und schreibt 1:1 Zuordnungen im Wagner-Format nach Stdout.
 * Sonstige Meldungen werden auf Stderr ausgegeben.
 * Idee und Algorithmus sind in der Dokumentation der Klasse
 * de.denkselbst.matrixAligner.BestCellMatrixAligner beschrieben.
 */
public class HeadingMapper4Arno{
    public static void main(String args[]) throws IOException{

        if(args.length > 0){
            System.err.println("HeadingMapper4Arno liest "+
                "Matrix von stdin, schreibt 1:1-Mapping "+
                "nach stdout");
            System.exit(1);
        }

        // Matrixobjekt anlegen
        Matrix matrix = new Matrix();

        // Alignerobjekt anlegen
        MatrixAligner aligner = new BestCellMatrixAligner(matrix);

        // besorge Matrix aus Stdin, Aligner bekommt die Matrixänderung
        // automatisch von der Matrix mitgeteilt (Observer-Modell)
        matrix.setzeDatenAusStdin();

        // Normalisiere Matrix
        matrix.normalisiereDich();

        // aligne
        Pfad p = aligner.align();

        // schreibe Ergebnis
        System.out.println(p.toWagnerString(matrix.holeQuelle(),matrix.holeInfo()));
    }
}

```

M.5.23 de/denkselfbst/kommandozeile/manifest

Main-Class: de.denkselfbst.kommandozeile.Align

M.5.24 de/denkselfbst/kommandozeile/manifest2

Main-Class: de.denkselfbst.kommandozeile.HeadingMapper4Arno

M.5.25 de.denkselfbst.beobachtung.Beobachter.java

```
package de.denkselfbst.beobachtung;

/**
 * Wenn ein beobachtetes Objekt eine Änderung zu melden hat, ruft es die in diesem
 * Interface deklarierte Methode bei seinen Beobachtern auf.
 */

public interface Beobachter{
    /**
     * Diese Methode wird von dem beobachteten Objekt ausgelöst, wenn
     * eine Änderung erfolgt ist. Das Objekt liefert sich selbst und
     * eine Nachricht (String) beim Aufruf mit.
     */
    public void beobachtetesObjektHatSichGeregt(Object o, String nachricht);
}
```

M.5.26 de.denkselfbst.beobachtung.Beobachtbar.java

```
package de.denkselfbst.beobachtung;

/**
 * Dieses Interface legt die Methoden fest, die ein Objekt implementieren muss, damit
 * es von anderen Objekten beobachtet werden kann.
 */

public interface Beobachtbar{

    /**
     * einen neuen Beobachter in die Liste der im Falle einer Änderung zu benachrichtigen
     * Objekte aufnehmen
     * @param b das Objekt, das Interesse an Änderungen hat
     */
    public void beobachterAnmelden (Beobachter b);

    /**
     * einen Beobachter aus der Liste der zu benachrichtigen
     * Objekte entfernen.
     * @param b das Objekt, das das Interesse an den Änderungen verloren hat
     */
    public void beobachterAbmelden (Beobachter b);
}
```

M.5.27 de.denkselfbst.mavis.Mavis.java

```
package de.denkselfbst.mavis;

import de.denkselfbst.matrix.Pfad;
import de.denkselfbst.matrix.KoksZuordnung;
import de.denkselfbst.matrix.Matrix;
import de.denkselfbst.matrix.MatrixView;
import de.denkselfbst.matrix.MatrixHistogramView;
```

```

import de.denkselbst.matrixAligner.MatrixAligner;
import de.denkselbst.matrixAligner.BFSMatrixAligner;

import java.io.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.JOptionPane;
import java.awt.event.ItemListener;
import java.awt.event.*;
import javax.swing.event.*;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JCheckBoxMenuItem;

/**
 * Diese Klasse setzt die grafische Oberfläche von Mavis zusammen und kümmert sich um Teile des Event-Handlings
 * Der Code ist immer mal wieder auf die Schnelle geändert worden. Er ist nicht besonders schön und auch mit de
 * heißen Nadel gestrickt. Aber das muss so sein, schließlich handelt es sich um einen Prototypen.
 */

public class Mavis extends JFrame implements ActionListener, ItemListener{

    /** Speichert aktuellen Dateinamen der Matrix*/
    private String filename;
    private MatrixView mv;
    private JScrollPane sp;
    private Matrix matrix;

    /** Referenz auf Histogramm-Fenster*/
    private JFrame hist;
    private MatrixHistogramView histView;

    /** enthält aktuellen Dateinamen..*/
    private JLabel la;

    //Create a file chooser
    private final JFileChooser fc = new JFileChooser();

    public Mavis(Matrix mat, MatrixView mv, String title){
        hist = null;
        histView = null;

        filename = title;
        this.mv = mv;
        this.matrix = mat;

        sp = new JScrollPane(mv, JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED, JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
        sp.setMinimumSize(new Dimension(320,200)); // Mindestgröße von 320 * 200 einfordern

        // MatrixView sagen, dass sie in dieser Scrollpane steckt! (für MatrixView.XYInfo)
        //mv.setzeScrollPane(sp);

        setTitle(title+" - Mavis");
        getContentPane().add(sp, BorderLayout.CENTER);

        // Menüs erstellen
        JRadioButtonMenuItem rbMenuItem;
        JCheckBoxMenuItem cbMenuItem;
        JMenuBar menuBar;

```

```
        JMenu menu, submenu;
        JMenuItem menuItem;
        menuBar = new JMenuBar();
        setJMenuBar(menuBar);

        menu = new JMenu("Matrix");
        menuBar.add(menu);
        menuItem = new JMenuItem("öffnen");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menuItem = new JMenuItem("zurücksetzen");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menu.addSeparator();

        menuItem = new JMenuItem("Normalisieren");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menuItem = new JMenuItem("RangfolgenClustering");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menuItem = new JMenuItem("stutzen");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menu.addSeparator();

        menuItem = new JMenuItem("Histogramm-Fenster");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menu = new JMenu("Alignment");
        menuBar.add(menu);
        menuItem = new JMenuItem("A*");
        menuItem.addActionListener(this);
        menu.add(menuItem);
        menuItem = new JMenuItem("Best First Search");
        menuItem.addActionListener(this);
        menu.add(menuItem);
        menuItem = new JMenuItem("Levenshtein Minimum Edit Distance");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menu.addSeparator();

        menuItem = new JMenuItem("Best Cell 1:1");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        menu.addSeparator();

        menuItem = new JMenuItem("selektiertes Alignment speichern");
        menuItem.addActionListener(this);
        menu.add(menuItem);
        menuItem = new JMenuItem("Alignment laden und anzeigen");
```

```
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("selektiertes Alignment textuell anzeigen");
menuItem.addActionListener(this);
menu.add(menuItem);

menu.addSeparator();

menuItem = new JMenuItem("selektiertes Alignment löschen");
menuItem.addActionListener(this);
menu.add(menuItem);
menuItem = new JMenuItem("Alle Alignments löschen");
menuItem.addActionListener(this);
menu.add(menuItem);

menu = new JMenu("Ansicht");
menuBar.add(menu);
submenu = new JMenu("Blockgröße");
menu.add(submenu);
menuItem = new JMenuItem("30");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("20");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("18");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("16");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("14");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("12");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("10");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("8");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("6");
submenu.add(menuItem);
menuItem.addActionListener(this);

menuItem = new JMenuItem("4");
submenu.add(menuItem);
menuItem.addActionListener(this);

menu.add(submenu);
```

```

cbMenuItem = new JCheckBoxMenuItem("Diagonale anzeigen");
cbMenuItem.addItemListener(this);

menu.add(cbMenuItem);

la = new JLabel(""+title);
getContentPane().add(la, BorderLayout.SOUTH);

JButton rl = new JButton("reload");
rl.addActionListener(this);
//getContentPane().add(rl, BorderLayout.SOUTH);

addWindowListener( new WindowAdapter(){
    public void windowClosing(WindowEvent we){
        setVisible(false);
        dispose();
        System.exit(0);
    }
}

);

pack();
setLocation(220,100);
show();
}

public void actionPerformed(ActionEvent e){
    String cmd = e.getActionCommand();

    /** Hier muss unbedingt aufgeräumt werden if-else if-else if, bzw
        innere Klassen, ...
        (Sobald mal feststeht, was alles wie aussehen soll...)
        */

    System.err.println("Action: "+cmd);

    if (cmd.equals("zurücksetzen")){
        if(matrix!=null) matrix.setzeDatenAusDatei(filename);
    }

    if(cmd.equals("öffnen")){
        int returnVal = fc.showOpenDialog(this);
        if (returnVal == JFileChooser.APPROVE_OPTION) {
            File file = fc.getSelectedFile();
            System.err.println("Opening: " + file.getName() + ".");

            if(matrix!=null) matrix.setzeDatenAusDatei(file.toString());
            else{
                matrix = new Matrix(file.toString());
                mv.setzeMatrix(matrix);
            }

            setTitle(file.toString()+" - Mavis");
            la.setText(file.toString());

```

```

filename = file.toString(); // Dateinamen merken...
if(hist != null) hist.setTitle("Histogramm - "+file.toString()+" - Mavis");
    // alte Alignmentpfade wegbürsten
    mv.loescheAllePfade();

}else {
    System.err.println("Open command cancelled by user." );
}
}

if(cmd.equals("RangfolgenClustering")){
    String inputValue = JOptionPane.showInputDialog(this, "Wie viele Klassen?");
    int val;
    if(inputValue == null) return;
    try{
        val = Integer.parseInt(inputValue);
    }catch(NumberFormatException nmfe){
        JOptionPane.showMessageDialog(this, "Konnte die Eingabe "+inputValue+" nicht als Zahl interpretieren");
        val = 10;
    }
    val = Math.abs(val);
    matrix.rangfolgenClustering(val);
}

if(cmd.equals("Normalisieren")){
    matrix.normalisiereDich();
}

if(cmd.equals("stutzen")){
    StutzenDialog s = new StutzenDialog(this, matrix.holeBreite(), matrix.holeHoehe());
    if(s.cancelled()!=true){
        int xn = s.getXVal();
        int yn = s.getYVal();
        matrix.stutzenAuf(xn,yn);
        // alte Alignmentpfade wegbürsten
        mv.loescheAllePfade();
    }
    s.dispose();
}

if(cmd.equals("Alle Alignments löschen")){
    mv.loescheAllePfade();
}

if(cmd.equals("selektiertes Alignment löschen")){
    mv.entferneSelektiertenAlignmentPfad();
}

if(cmd.equals("selektiertes Alignment textuell anzeigen")){
    mv.zeigeSelektiertenAlignmentPfadAlsHTML();
}

if(cmd.equals("selektiertes Alignment speichern")){

    // Checken, ob was selektiert ist!!!
    if (!mv.istEtwasSelektiert()) return;

    JFileChooser fc = new JFileChooser();

```

```
int returnVal = fc.showSaveDialog(this);

if (returnVal == JFileChooser.APPROVE_OPTION) {
    File file = fc.getSelectedFile();
    //this is where a real application would save the file.
    mv.speichereSelektiertenAlignmentPfad(file);

    System.err.println("Pfad gespeichert in "+file.toString());
} else {
    System.err.println("Pfad Speichern abgebrochen...");
}

}

if(cmd.equals("Alignment laden und anzeigen")){
    int returnVal = fc.showOpenDialog(this);
    if (returnVal == JFileChooser.APPROVE_OPTION) {
        File file = fc.getSelectedFile();
        System.err.println("Opening: " + file.getName() + ".");

        KoksZuordnung al = new KoksZuordnung(file.toString());

        mv.fuegeKoksZuordnungEin(al,file.toString() );

    }else {
        System.err.println("Open command cancelled by user.");
    }
}

// verschiedene Alignments abfahren
//
if(cmd.equals("A*")){
    MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.AStarMatrixAligner");
    macv = null; // Ist das Ok? Wenn ja warum? Wenn nein, warum nicht? Wartet macv im Konstruktor, bis a
}

if(cmd.equals("Best First Search")){
    MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.BFSMatrixAligner");
    macv = null;
}

if(cmd.equals("Levenshtein Minimum Edit Distance")){
    MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.LMEDMatrixAligner");
    macv = null;
}

if(cmd.equals("Best Cell 1:1")){
    MACV macv = new MACV(matrix, mv, "de.denkselbst.matrixAligner.BestCellMatrixAligner");
    macv = null;
}
```



```

if(cmd.equals("30")){mv.setzeBlockGroesse(30,30);}
if(cmd.equals("20")){mv.setzeBlockGroesse(20,20);}

if(cmd.equals("18")){mv.setzeBlockGroesse(18,18);}
if(cmd.equals("16")){mv.setzeBlockGroesse(16,16);}
if(cmd.equals("14")){mv.setzeBlockGroesse(14,14);}
if(cmd.equals("12")){mv.setzeBlockGroesse(12,12);}

if(cmd.equals("10")){mv.setzeBlockGroesse(10,10);}
if(cmd.equals("8")){mv.setzeBlockGroesse(8,8);}
if(cmd.equals("6")){mv.setzeBlockGroesse(6,6);}
if(cmd.equals("4")){mv.setzeBlockGroesse(4,4);}

if(cmd.equals("Histogramm-Fenster")){
    //Ist schon eins offen?
    if(histView != null){

        // Aufhören, die Matrix zu beobachten!
        histView.abmelden();
        //
        // Fenster wegwerfen
        hist.dispose();
    }
    //nein, dann neu

    /* Histogramm-Fenster */
    // Bastele Histogramm-Fenster
    hist = new JFrame("Histogramm - "+filename+" - Mavis");
    histView = new MatrixHistogramView(matrix);
    hist.getContentPane().add( histView, BorderLayout.CENTER );
    hist.setSize(new Dimension(220,70));
    hist.setResizable(false);
    hist.setLocation(0,300);

    hist.addWindowListener( new WindowAdapter(){
        public void windowClosing(WindowEvent event){
            histView.abmelden();
            histView = null;
            hist.setVisible(false);
            hist.dispose();
        }
    }
    );

    // Radiobuttons einfügen
    JRadioButton sg02 = new JRadioButton("05");
    sg02.setActionCommand("0.2");
    JRadioButton sg01 = new JRadioButton("10");
    sg01.setActionCommand("0.1");
    JRadioButton sg005 = new JRadioButton("20");
    sg005.setActionCommand("0.05");
    sg005.setSelected(true);
    JRadioButton sg0025 = new JRadioButton("40");
    sg0025.setActionCommand("0.025");
    JRadioButton sg00125 = new JRadioButton("80");
    sg00125.setActionCommand("0.0125");
    // Group the radio buttons.

```

```

    ButtonGroup group = new ButtonGroup();
    group.add(sg02);
    group.add(sg01);
    group.add(sg005);
    group.add(sg0025);
    group.add(sg00125);

    // Register a listener for the radio buttons.
    /** Listens to the radio buttons. */
    class RadioListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {

        histView.setzeSchrittgroesse(Double.parseDouble(e.getActionCommand()));

    }
    }

    RadioListener myListener = new RadioListener();
    sg02.addActionListener(myListener);
    sg01.addActionListener(myListener);
    sg005.addActionListener(myListener);
    sg0025.addActionListener(myListener);
    sg00125.addActionListener(myListener);

    // Put the radio buttons in a column in a panel
    JPanel radioPanel = new JPanel();
    radioPanel.setLayout(new GridLayout(0,1));
    radioPanel.add(sg02);
    radioPanel.add(sg01);
    radioPanel.add(sg005);
    radioPanel.add(sg0025);
    radioPanel.add(sg00125);
    //setLayout(new BorderLayout());
    hist.getContentPane().add(radioPanel, BorderLayout.WEST);

    hist.show();

}

}

public void itemStateChanged(ItemEvent e){
// jaja, ist schlecht, da eine zweite Checkbox nicht von der ersten unterschieden werden kann...
System.err.println("ItemEvent: "+e.getStateChange());
    if(e.getStateChange()==ItemEvent.SELECTED){
        mv.zeigeDiagonale(true);
    }else{
        mv.zeigeDiagonale(false);
    }
}

/** Main-Methode bringt den Stein ins Rollen. */
public static void main(String[] args) throws IOException{

//System.err.println(""+System.getProperty("user.dir"));
//System.err.println(""+System.getProperty("java.class.path"));

```

```

String title = "[keine Datei geladen]" ;

Matrix matrix = null;
// Besorge Matrix aus Datei
if(args.length > 0){
    matrix = new Matrix(args[0]);
    title = args[0];
}else{
//    matrix = new Matrix(10,10);
}

// Bastele MatrixView dazu
//MatrixView mv = new MatrixView(matrix);
MatrixView mv = new MatrixView();
mv.setzeMatrix(matrix);

// Bastele Mavis dazu
Mavis m = new Mavis(matrix, mv, title);

}

}

```

M.5.28 de.denkselbst.mavis.MACV.java

```

package de.denkselbst.mavis;

import de.denkselbst.matrix.Matrix;
import de.denkselbst.matrix.Pfad;
import de.denkselbst.matrix.MatrixView;
import de.denkselbst.matrixAligner.MatrixAligner;
import de.denkselbst.matrixAligner.AlignmentObserver;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Dimension;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;

import java.lang.reflect.Constructor;
import java.lang.reflect.InvocationTargetException;
/**
 * Objekte dieser Klasse rufen einen MatrixAligner ins Leben und beobachten ihn.
 * Die Nachrichten des Aligners werden in einem eigenen Fenster dargestellt,
 * das von dieser Klasse erzeugt wird.
 * Objekte dieser Klasse werden von Mavis.java ins Leben gerufen, wenn der
 * Benutzer ein Alignment veranlasst.
 */
public class MACV extends JFrame implements ActionListener, AlignmentObserver{

```

```

private Matrix m;
private MatrixAligner aligner;

private JLabel status;
private JButton cancelButton;
private JButton okButton;
private MatrixView mv;

/**
 * Konstruktor.
 * @param m Matrix, die die Daten zum Alignen liefert
 * @param mv MatrixView, die das Ergebnis bekommen soll
 * @param klasse Klasse, von der ein MatrixAligner instantiiert werden soll
 * (Z.B. de.denkselbst.matrixAligner.BFSMatrixAligner, de.denkselbst.matrixAligner.LMEDMatrixAligner,
 * de.denkselbst.matrixAligner.AStarMatrixAligner...)
 */
public MACV(Matrix m, MatrixView mv, String klasse){

    boolean ok = true;           // Flagge, ob nachfolgende Reflection-Aktionen geklappt haben und
                                // ein MatrixAligner instantiiert werden konnte

    // von MatrixAligner abstammendes Objekt der Klasse klasse anlegen
    // bzw. versuchen, es anzulegen
    try{
        // Wenn Matrix m == null, dann können wir uns das alles schenken...
        if(m == null){
            throw new Exception("Keine Matrix vorhanden! Laden Sie eine!");
        }

        // Klasse zum String klasse besorgen
        Class alignClass = Class.forName(klasse);

        // formale Parameter erzeugen
        Class parameter[] = new Class[1];
        parameter[0] = m.getClass();
        // passenden Konstruktor besorgen
        Constructor alignConstructor = alignClass.getConstructor(parameter);

        // Argumente für den Konstruktoraufruf erzeugen
        Object argumente[] = new Object[1];
        argumente[0] = m;
        // Konstruktor mit Argumenten aufrufen und damit MatrixAligner-Objekt erzeugen
        aligner = (MatrixAligner) alignConstructor.newInstance(argumente);

    } // Mögliche Exceptions behandeln
    catch(ClassNotFoundException e){
        ok = false;
        System.err.println(e.getMessage());
        e.printStackTrace();
    }catch(NoSuchMethodException e){
        ok = false;
        System.err.println(e.getMessage());
        e.printStackTrace();
    }catch(InstantiationException e){
        ok = false;
        System.err.println(e.getMessage());
        e.printStackTrace();
    }catch(IllegalAccessException e){
        ok = false;

```

```

    System.err.println(e.getMessage());
    e.printStackTrace();
} catch (IllegalArgumentException e) {
    ok = false;
    System.err.println(e.getMessage());
    e.printStackTrace();
} catch (InvocationTargetException e) {
    ok = false;
    System.err.println(e.getMessage());
    e.printStackTrace();
} catch (Exception e) {
    ok = false;
    System.err.println(e.getMessage());
}

} finally {
    if (!ok) {
        System.err.println("Es gab bei dem Versuch ein MatrixAligner Objekt"+
            " der Klasse "+klasse+" anzulegen eine Exception. (siehe oben)");
    } else {
        System.err.println("MatrixAligner der Klasse "+klasse+" angelegt.");
    }
}

this.mv = mv; // MatrixView merken
this.m = m; // Matrix merken

JLabel label = new JLabel(); // Label anlegen
status = new JLabel(); // Status-Label anlegen
cancelButton = new JButton("Abbrechen!"); // Abbrechen-Knopf anlegen
cancelButton.setEnabled(true);
cancelButton.addActionListener(this);
okButton = new JButton("OK"); // Ok-Knopf anlegen
okButton.setEnabled(false);
okButton.addActionListener(this);

if (ok && (m != null)) { // Aligner sowie Matrix vorhanden
    String temp = aligner.alignerInfo();
    setTitle(temp); // Fenstertitel setzen
    label.setText("Hier arbeitet ein "+temp+"."); // Label setzen
    status.setText("Starte...");
}
else { // kein Aligner vorhanden
    setTitle("Achtung!"); // Fenstertitel setzen
    label.setText("Der Computer hat die Arbeit niedergelegt.");

    if (m == null) status.setText("Grund: Es ist noch keine Matrix geladen worden.");
    else status.setText("Grund: Das MatrixAligner-Objekt konnte nicht angelegt werden.");

    cancelButton.setText("Was solls!");
}

// GUI-Objekte in das Frame einhängen
getContentPane().add(label, BorderLayout.NORTH);
getContentPane().add(status, BorderLayout.CENTER);
JPanel btnPanel = new JPanel(new FlowLayout());
btnPanel.add(cancelButton);
btnPanel.add(okButton);
getContentPane().add(btnPanel, BorderLayout.SOUTH);

pack(); // Layout zusammenschieben
show(); // Fenster anzeigen

```

```

// nur wenn alles OK ist, es also einen Aligner gibt, wird mit uns als observer alignt.
if(ok){
    aligner.konfigurationsDialogAbfahren();           // Konfigurationsmethode anstossen
    aligner.align(this);                             // Alignen...
}
}

/** ActionListener */
public void actionPerformed(ActionEvent e){

    if(e.getSource() == okButton){
        if(aligner!=null)aligner.abmelden();         // Meldet den Aligner von der Matrix ab,
        aligner = null;                             // Frei zum garbage-collecten
        dispose();                                   // damit keine Referenzen mehr auf ihn existieren und er
    }                                                 // garbage-collected werden kann
    if(e.getSource() == cancelButton){
        if(aligner!=null){
            aligner.cancel();                         // "
            aligner.abmelden();                       // "
            aligner = null;                           // Frei zum garbage-collecten
        }
        dispose();
    }
}

/** AlignmentObserver */
public void alignmentFertig(Pfad p, boolean erfolg){
    okButton.setEnabled(true);
    cancelButton.setEnabled(false);
    if(erfolg){
        mv.fuegeAlignmentPfadHinzu(p,aligner.alignerInfo());
        System.out.println("\nAlignment-Pfad:");
        System.out.println(p.toString());
        System.out.println("\nWagner-Format:");
        System.out.println(p.toWagnerString(m.holeQuelle(),m.holeInfo()));
    }
}

/** AlignmentObserver */
public void statusInformation(String s, boolean erfolg){
    status.setText(s);
}
}

```

M.5.29 de.denkselbst.mavis.StutzenDialog.java

```

package de.denkselbst.mavis;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import javax.swing.JFrame;

```

```

import javax.swing.JDialog;
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JButton;
import javax.swing.JSlider;
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.BorderLayout;
import java.awt.GridLayout;

/**
 * Definiert einen Dialog, mit dem eine Matrix verkleinert werden kann.
 * Wird von Mavis.java aus ins Leben gerufen.
 */
public class StutzenDialog extends JDialog implements ActionListener, ChangeListener{

    private JSlider x;
    private JLabel xl;
    private JSlider y;
    private JLabel yl;
    private JButton ok;
    private JButton cancel;
    private boolean cancelled = false;

    public StutzenDialog(JFrame parent, int maxx, int maxy){
        super(parent, "Matrix verkleinern", true);

        JPanel panel = new JPanel(new GridLayout(2,2));

        x = new JSlider(0,maxx,maxx); x.setSnapToTicks(true);
        y = new JSlider(0,maxy,maxy); y.setSnapToTicks(true);
        x.addChangeListener(this);
        y.addChangeListener(this);
        xl = new JLabel("Breite: "+maxx);
        panel.add(xl);
        panel.add(x);

        yl = new JLabel("Höhe: "+maxy);
        panel.add(yl);
        panel.add(y);

        getContentPane().add(panel, BorderLayout.CENTER);

        ok = new JButton("ok");
        ok.addActionListener(this);
        cancel = new JButton("abbrechen");
        cancel.addActionListener(this);
        JPanel btnPanel = new JPanel(new GridLayout(2,2));
        btnPanel.add(ok);
        btnPanel.add(cancel);

        getContentPane().add(btnPanel, BorderLayout.SOUTH);

        pack();
        show();
    }

    // ActionListener

```

```

public void actionPerformed(ActionEvent e){
    if(e.getSource() == ok){
        setVisible(false);
    }
    if(e.getSource() == cancel){
        setVisible(false);
        cancelled = true;
    }
}

// ChangeListener
public void stateChanged(ChangeEvent e){
    xl.setText("Breite: "+getXVal());
    yl.setText("Höhe: "+getYVal());
}

// x-Wert des Sliders liefern
public int getXVal(){
    return x.getValue();
}

// y-Wert des Sliders liefern
public int getYVal(){
    return y.getValue();
}

public boolean cancelled(){
    return cancelled;
}

/** Kleiner Test / Demo */
public static void main (String[] args){
    StutzenDialog s = new StutzenDialog(null, 20,7);

    if(!s.cancelled()){
        System.out.println("x: "+s.getXVal()+" y: "+s.getYVal());
    }
}
}

```

M.5.30 de.denkselbst.mavis.ShowHTMLFrame.java

```
package de.denkselbst.mavis;
```

```

import javax.swing.JFrame;
import javax.swing.JEditorPane;
import javax.swing.JScrollPane;
import java.awt.event.WindowListener;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.Dimension;

```

```

/**
 * Diese Klasse öffnet ein HTML-Fenster auf der Oberfläche, das per Konstruktor mit HTML-Code gefüllt wird.
 * Zweck ist das Anzeigen eine Alignments in Textform.
 * Das Fenster ist nach seiner Erzeugung unabhängig von anderen Programmteilen und wird von keinem
 * Objekt kontrolliert. Es reagiert lediglich auf das Schließen-Event, das vom Schließen-Knopf im
 * Fensterrahmen gesendet wird.

```



```
*/
public class ShowHTMLFrame extends JFrame{
    /** Konstruktor */
    public ShowHTMLFrame(String title, String text){
        super(title);

        // JEditorPane zum Anzeigen von HTML benutzen
        JEditorPane htmlpane = new JEditorPane(); // Editor anlegen
        htmlpane.setEditable(false);             // nicht ditierbar
        htmlpane.setContentType("text/html");    // Inhalt = html
        htmlpane.setText(text);                  // Text einsetzen
        htmlpane.setPreferredSize(new Dimension(600,400));

        // Scrollpane um htmlpane wickeln und ins Frame packen
        JScrollPane sp = new JScrollPane(htmlpane);
        sp.setPreferredSize(new Dimension(600,400));
        getContentPane().add(sp);

        // Listener zum Schließen des Fensters
        addWindowListener( new WindowAdapter(){
            public void windowClosing(WindowEvent we){
                setVisible(false);
                dispose();
            }
        });

        pack();
        show();
    }

    /**
     * Kleiner Test
     */
    public static void main(String[] args){
        ShowHTMLFrame sf = new ShowHTMLFrame("TestHTML", "<h1>foobar</h1>");
    }
}
```

M.5.31 de/denkselfbst/mavis/manifest

Main-Class: de.denkselbst.mavis.Mavis

Anhang N

Phrasenalignment – Code

N.1 Python-Paket Phrase

N.1.1 `__init__.py`

```
1  #!/usr/bin/python
2
3  # Initialisierung des Pakets "Region"
4
5  __all__ = [
6      "zuordnung.py",
7  ]
```

N.1.2 `getsegs.py`

Sucht Segmente einer bestimmten Länge in der Datenbank.

```
1  #!/usr/bin/python
2
3  import sys, string
4  from DatabaseAPI import db, config, segmentdb
5
6  def get_segs(dbconn, limit, korpora):
7      list = dbconn.executelist("select m.segmentnr from map_s_w m, saetze s where m.satzid = s.satzid and s.he
8      if list:
9          list = dbconn.executelist("select segmentnr, count(tokenid) from map_s_w where segmentnr in (%s) grou
10         return list
11         return None
12
13  def printify(list):
14      trans = string.maketrans("", ",")
15      return string.translate('list', trans, "([L,")[:-2]
16
17
18  def print_segs(list, dbconn, file):
19      textfile = open(file + ".txt", "w")
20      segfile = open(file + ".segs", "w")
21      # print list
22      for id in list:
23          segid = db.unknown2string(id[2])
24          segfile.write("%s\n" %segid)
25          textfile.write("%s:\n" %segid)
26          for lang in ("1", "2"):
27              segment = segmentdb.SegmentVonDb(segid, dbconn, lang)
28              tags = segment.getTags()
29              words = segment.getWoerter()
30              for(wort, tag) in map(None, words, tags):
```

```

31         textfile.write("%s\t%s\n" %(wort, tag))
32         if lang == "1":
33             textfile.write(10 * "*" + "\n")
34         textfile.write(30 * "-" + "\n")
35
36 def order_segs(list, count):
37     idlist = []
38     sum = 0
39     segid, swordcount = list[0]
40     for row in list:
41         (nsegid, wordcount) = row
42         if nsegid == segid:
43             sum = sum + wordcount
44             if swordcount > wordcount:
45                 swordcount = wordcount
46         else:
47             idlist.append((swordcount, sum, segid))
48             segid, sum = nsegid, wordcount
49             swordcount = sum
50     idlist.append((swordcount, sum, segid))
51     idlist.sort()
52     return idlist[0:count]
53
54
55 def main():
56     usage = sys.argv[0] + " <segmentgroesse> <segmentanzahl> wb|ko <filename>"
57     if len(sys.argv) < 5 or not sys.argv[3] in ("wb", "ko"):
58         print usage
59         sys.exit(1)
60     korpora = " in %s " %config.wbids
61     if sys.argv[3] == "ko":
62         korpora = " in %s " %config.goodcorpora
63     #     korpora = " not " + korpora
64     file = sys.argv[4]
65     dbconn = db.Db()
66     list = get_segs(dbconn, sys.argv[1], korpora)
67     #     print list
68     if list:
69         print_segs(order_segs(list, int(sys.argv[2])), dbconn, file)
70
71 if __name__ == '__main__':
72     main()
73

```

N.1.3 zuordnung2.py

Versucht, korrespondierende Phrasen zu finden.

```

1  #!/usr/bin/python
2
3  import string
4  import StringIO
5  import zuordnung, wb, filterwb
6  from DatabaseAPI import config
7  from Region import atom, satz
8
9  class OrdneZu:
10
11     def __init__(self, lang1, lang2):
12         self.lang1 = lang1
13         self.lang2 = lang2

```

```

14     self.wb = {}
15     self.wb[self.lang1] = wb.LemmatabuchVonDB(self.lang1, self.lang2)
16     self.wb[self.lang2] = wb.LemmatabuchVonDB(self.lang2, self.lang1)
17     self.tags= {}
18     self.tags[self.lang1] = filterwb.tags[config.lang2id[self.lang1]]
19     self.tags[self.lang2] = filterwb.tags[config.lang2id[self.lang2]]
20     self.tagfolgen = {}
21     self.tagfolgen[self.lang1] = self.loadTags(self.lang1, config.tags)
22     self.tagfolgen[self.lang2] = self.loadTags(self.lang2, config.tags)
23     self.info = StringIO.StringIO()
24
25 def loadTags(self, lang, dir):
26     file = open("%s/tags_%s.txt" %(dir, lang), "r")
27     tagfolgen = {}
28     cat = ""
29     hash = {}
30     for line in file.readlines():
31         line = string.strip(line)
32         pos = string.find(line, "###**")
33         if pos > -1:
34             if cat:
35                 tagfolgen[cat] = hash
36                 cat = line[pos + 6:-1]
37                 hash = {}
38                 continue
39             if not cat:
40                 continue
41             tags = string.split(line, " ")
42             for tag in tags:
43                 arr = []
44                 if hash.has_key(tag):
45                     arr = hash[tag]
46                 arr.append(line)
47                 hash[tag] = arr
48             tagfolgen[cat] = hash
49             file.close()
50             return tagfolgen
51
52 def getInfo(self):
53     self.info.seek(0)
54     return self.info.read()
55
56 def ordne_zu(self, phrasel, phrase2):
57     self.info = StringIO.StringIO()
58     ph1 = phrasel.getRegionenMit(atom.Atom)
59     ph2 = phrase2.getRegionenMit(atom.Atom)
60     kristalle1, kristalle2 = self.dropKnown(ph1, self.lang1, ph2, self.lang2)
61     tagfolgen1, rest1 = self.findTagfolgen(kristalle1, self.lang1, ph1)
62     tagfolgen2, rest2 = self.findTagfolgen(kristalle2, self.lang2, ph2)
63     phrasen = self.ordneTagfolgenZu(tagfolgen1, tagfolgen2)
64     phrasen.append((rest1, rest2))
65     return phrasen
66
67 def dropKnown(self, phrasel, lang1, phrase2, lang2):
68     # phrasel & 2 sind jeweils Regionen mit Atom
69     crystals1, crystals2 = [], [] # Liste von (index, atom, crystal) Paaren
70     #es ist markiert, ob etwas uebersetzbar war oder nicht - nur unuebersetzbaren wird zugeordnet
71     found1, found2 = {}, {}
72     for (lang, phrase, crystals, otherphrase, found, ofound) in [
73         (lang1, phrasel, crystals1, phrase2, found1, found2),
74         (lang2, phrase2, crystals2, phrasel, found2, found1)

```

```

75 ]:
76     wb = self.wb[lang]
77     irrel = self.tags[lang]          # Bewertung der Einzeltags
78     for i in range(len(phrase)):
79         patom = phrase[i]
80         hypo = (i, patom, 0)
81         uehypo = (i, patom, 1)
82         tag = patom.getTags()[0]
83         lemma = patom.getLemmata()
84         wort = patom.getWoerter()[0]
85         if found.has_key(wort):
86             crystals.append(uehypo)
87         if irrel.has_key(tag) and irrel[tag] < 1:
88             continue
89         if lemma[0] == "<unknown>":
90             if "<unknown>" in otherphrase.getLemmata() and wort in otherphrase.getWoerter():
91                 # warum reicht nicht die zweite Bedingung?
92                 # weil das Wort in der andere Phrase womoeglich nicht unknown ist...
93                 # lass uns aber nochmal drueber reden!
94                 ofound[word] = "found"
95                 continue
96         if not wb.has_key(lemma):
97             crystals.append(hypo)
98             continue
99         uebers = wb[lemma]
100        if len(uebers) == 0:
101            crystals.append(hypo)
102            continue
103        else:
104            wfound = 0
105            otherlems = string.join(otherphrase.getLemmata())
106            for (x, eintrag, uebliste) in uebers:
107                if len(eintrag.getWoerter()) > 1:
108                    continue
109                for ueb in uebliste:
110                    uelems = string.strip(string.join(ueb.getLemmata()))
111                    #todo: Wortgrenzen!!!
112                    if uelems and string.find(otherlems, uelems) > -1:
113                        ofound[ueb.getWoerter()[0]] = "found"
114                        wfound = 1
115                        self.info.write("%s -> %s\n" % ('lemma', 'ueb.getLemmata()'))
116                        break
117                if not wfound:
118                    crystals.append(hypo)
119            else:
120                crystals.append(uehypo)
121        return crystals1, crystals2
122
123    def p_findTagfolge(self, tagfolge, alltags, pos):
124        #anzahl der leerzeichen in alltags bis pos = phrasenanfang
125        #anzahl der leerzeichen in tagfolge + start = phrasenende
126        start = string.count(alltags, " ", 0, pos)
127        end = start + string.count(tagfolge, " ") + 1
128        return (start,end)
129
130    def findTagfolgen(self, kristall, lang, phrase):
131        tagfolgen = {}
132        rest = []
133        ourtags = self.tagfolgen[lang]
134        slices = {}
135        for (pos, patom, ueb) in kristall:

```

```

136         found = 0
137         tag = patom.getTags()[0]
138         min, max = pos - 6, pos + 6
139         if min < 0:
140             min = 0
141         if max >= len(phrase):
142             max = len(phrase) - 1
143         tagrel = " " + string.join(phrase[min:max].getTags()) + " "
144         for kat in ourtags.keys():
145             if not ourtags[kat].has_key(tag):
146                 continue
147             poss_tags = ourtags[kat][tag]
148             for ptag in poss_tags:
149                 tpos = string.find(tagrel, " " + ptag + " ")
150                 if tpos > -1:
151                     arr = []
152                     if tagfolgen.has_key(kat):
153                         arr = tagfolgen[kat]
154                         start, end = self.p_findTagfolge(ptag, tagrel, tpos)
155                         if end < pos or start > pos or slices.has_key((start,end)):
156                             continue
157                         found = 1
158                         start = start + min
159                         end = end + min
160                         arr.append((phrase[start:end], ueb))
161                         slices[(start,end)] = "found" # eine Tagfolge in 2 Kats???
162                         tagfolgen[kat] = arr
163             if not found:
164                 rest.append(patom)
165         return tagfolgen, satz.Satz(rest)
166
167     def ordneTagfolgenZu(self, folge1, folge2):
168         phrasen = []
169         for kat in folge1.keys():
170             if folge2.has_key(kat):
171                 phrasen = phrasen + self.multiply(folge1[kat], folge2[kat])
172         return phrasen
173
174     def multiply(self, list1, list2):
175         nlist = []
176         for (i, ueb1) in list1:
177             for (j, ueb2) in list2:
178                 if not (ueb1 and ueb2): #es darf nicht beides bekannt sein
179                     nlist.append((i, j))
180         return nlist
181

```

N.1.4 tags_de.txt

Deutsche Tagfolgen

```

1 #####ADVP:
2 ADV
3 ADJD
4 ADV ADV
5 ADV ADJD
6 ADJD ADJD
7 ADJD ADV
8 ADV ADV ADJD
9 ADV ADV ADV
10 PTKA ADJD

```

```

11 ADV ADJD ADJD
12 ADJD ADV ADV
13 ADJD ADV ADJD
14 ADV ADV ADV ADJD
15 ADV ADJD ADV
16 ###**AP:
17 ADJA
18 PIDAT
19 ADV ADJA
20 ADJD ADJA
21 ADV PIDAT
22 ADV ADV ADJA
23 APPR ART NN ADJA
24 APPRART NN ADJA
25 APPR NN ADJA
26 ADV ADJD ADJA
27 APPR NE ADJA
28 ADJD ADJD ADJA
29 PAV ADJA
30 APPR ADJA NN ADJA
31 ###**NP:
32 ART NN
33 ADJA NN
34 PPER
35 ART ADJA NN
36 PRF
37 PRELS
38 PIS
39 PPOSAT NN
40 ART NN ART NN
41 PDS
42 PDAT NN
43 PIAT NN
44 ART NE
45 PIDAT NN
46 ART NN NE
47 ART NN ART ADJA NN
48 PPOSAT ADJA NN
49 PWS
50 ART ADJA NN ART NN
51 ADV ADJA NN
52 ADJA NE
53 ART ADJA ADJA NN
54 ADJA NN CARD
55 ART CARD NN
56 ADJA ADJA NN
57 ADJA NN ART NN
58 ART ADJA NN NE
59 ADJD ADJA NN
60 ADJA NN NE
61 PIAT ADJA NN
62 ART PIDAT NN
63 ART NN NE NE
64 ART ADV ADJA NN
65 ART ADJD ADJA NN
66 NN
67 ART NN KON ART NN
68 ART TRUNC NN
69 ART TRUNC KON NN
70 ART ADJA NE
71 ART ADJA NN NE NE

```

72 ADJA NE NE
73 PRELAT NN
74 ART ADJA NN ART ADJA NN
75 ADJA KON ADJA NN
76 ART NN PPOSAT NN
77 ART NN ADJA NN
78 PDAT ADJA NN
79 ART NN ART NE
80 ART NN CARD
81 ART NN ART NN NE
82 ART NE NE
83 ART ADJA CARD NN
84 ADV PIDAT NN
85 PIDAT ADJA NN
86 ADJA NN KON ADJA NN
87 ART ADJA KON ADJA NN
88 ADV ADV ADJA NN
89 ADJA NN ART ADJA NN
90 ADJA NN NE NE
91 ART NN PDAT NN
92 ADJA NN ADJA NN
93 PPOSAT NN NE
94 PWAT NN
95 PPOSAT NN NE NE
96 ART ADV CARD NN
97 ART NN KON ART ADJA NN
98 NE
99 ART NN ART NN ART NN
100 ART ADJA TRUNC KON NN
101 ART APPR ART NN ADJA NN
102 ADJA TRUNC KON NN
103 ART ADJA NN KON ART ADJA NN
104 ART NN NN
105 ART APPR NN ADJA NN
106 ART APPRART NN ADJA NN
107 ###**PP:
108 APPR ART NN
109 APPRART NN
110 APPR NN
111 APPR NE
112 APPR ART ADJA NN
113 PAV
114 APPR ADJA NN
115 APPRART ADJA NN
116 APPR PPOSAT NN
117 APPR ART NN ART NN
118 APPR NN KON NN
119 APPR PDAT NN
120 APPRART NN ART NN
121 PWAV
122 APPR PRELS
123 APPR NE NE
124 APPR PIS
125 APPR ART NE
126 APPR CARD
127 APPR PIDAT NN
128 APPR NN ART NN
129 APPR PPER
130 APPR ART NN ART ADJA NN
131 APPR ART NN NE
132 APPR PPOSAT ADJA NN

133 APPRART NN NE
134 APPRART NE
135 APPR NN NE
136 APPR PIAT NN
137 APPR CARD NN
138 APPR ART ADJA NN ART NN
139 APPR PRF
140 APPRART NN ART ADJA NN
141 APPR NE KON NE
142 APPRART NN CARD
143 APPR ART CARD NN
144 APPR ART ADJA ADJA NN
145 APPR NN ART ADJA NN
146 APPR ART NN CARD
147 APPRART NN ADV
148 APPR TRUNC KON NN
149 APPR NN CARD
150 APPR NN NE NE
151 APPR ART ADJA NN NE
152 APPR ART ADV ADJA NN
153 APPR ADJA ADJA NN
154 APPR ADJA NN KON NN
155 APPR ART ADJD ADJA NN
156 APPRART ADJA NN CARD
157 APPR ART PIDAT NN
158 APPRART ADJA NN ART NN
159 APPR ADV ADJA NN
160 APPR ART ADJA CARD NN
161 APPR ART TRUNC NN
162 APPR ART NN KON NN
163 APPRART ADJA NE
164 APPR ADJD ADJA NN
165 APPR NN KON ADJA NN
166 APPR ART NN KON ART NN
167 APPRART NN KON NN
168 APPR ART ADJA NE
169 APPR ADJA KON ADJA NN
170 APPR ART TRUNC KON NN
171 APPR PDAT ADJA NN
172 APPR ART NN PPOSAT NN
173 APPR ART ADJA NN ART ADJA NN
174 APPR ADJA NN ART NN
175 APPRART PIS
176 APPRART ADJA NN NE
177 APPR ART NN NE NE
178 APPR PRELAT NN
179 APPR ADJA NN KON ADJA NN
180 APPR ART NN ART NE
181 APPR PDS
182 APPR ART NN APPRART NN
183 APPRART TRUNC KON NN
184 APPR PIDAT ADJA NN
185 APPRART NN PPOSAT NN
186 APPRART TRUNC NN
187 ###**VC:
188 VVFIN
189 VAFIN
190 VVPP
191 VVINF
192 VMFIN
193 VVPP VAFIN

```

194 PTKZU VVINF
195 VVINF VMFIN
196 VVPP VAINF
197 VVIZU
198 VVIMP
199 VVPP VAPP
200 VVPP VAINF VMFIN
201 VAINF
202 VAFIN VVPP
203 VAPP
204 VVINF VMINF
205 VVINF VAFIN
206 PTKZU VVINF VAFIN
207 VVPP VAPP VAFIN
208 VVINF VVINF
209 VVINF PTKZU VMINF
210 VVINF VVFIN
211 VAINF VMFIN
212 PTKZU VAINF
213 VVPP PTKZU VAINF
214 VMFIN VVINF
215 VAPP VAFIN
216 PTKZU VVINF VVFIN
217 VVINF PTKZU VVINF
218 VMFIN VVPP VAINF
219 ###**VP:
220         VVFIN
221         VAFIN
222         VVPP
223         VVINF
224         VMFIN
225         ART NN VVFIN
226         VVFIN PPER
227         VVFIN ART NN
228         VVPP VAFIN
229         PTKZU VVINF
230         ART NN VAFIN
231         VVFIN PRF
232         VAFIN ART NN
233         PPER VVFIN
234         VAFIN PPER
235         ART NN VVPP
236         VVINF VMFIN
237         VVPP VAINF
238         PPER VAFIN
239         APPR ART NN VVFIN
240         ART ADJA NN VVFIN
241         ADJA NN VVFIN
242         VVIZU
243         ART NN VVINF
244         APPRART NN VVFIN
245         VMFIN ART NN
246         ART ADJA NN VAFIN
247         VVFIN ART ADJA NN
248         ART NN VMFIN
249         APPR ART NN VVPP
250         VMFIN PPER
251         APPR NN VVFIN
252         ADJA NN VAFIN
253         VAFIN PRF
254         PAV VVFIN

```

255 VVIMP
256 APPR ART NN VAFIN
257 PDS VVFIN
258 VVFIN APPR ART NN
259 PDS VAFIN
260 VAFIN ART ADJA NN
261 APPR NE VVFIN
262 VVPP VAPP
263 ART NN PTKZU VVINF
264 VVFIN APPRART NN
265 ART NN VVPP VAFIN
266 PPER VMFIN
267 VVPP VAINF VMFIN
268 VAINF
269 ART ADJA NN VVPP
270 ADJA NN VVPP
271 VAFIN APPR ART NN
272 APPR ART NN VVINF
273 VAFIN VVPP
274 VAFIN APPRART NN
275 APPR ART ADJA NN VVFIN
276 APPR NN VVPP
277 VMFIN PRF
278 PAV VAFIN
279 VVFIN APPR NN
280 APPRART NN VAFIN
281 ART NN ART NN VVFIN
282 APPRART NN VVPP
283 VVFIN PAV
284 VVFIN ADJA NN
285 VVFIN PIS
286 PPOSAT NN VVFIN
287 PIS VVFIN
288 APPR NE VAFIN
289 APPR NN VAFIN
290 APPR ADJA NN VVFIN
291 ART ADJA NN VVINF
292 VAPP
293 APPR NN VVINF
294 PDAT NN VVFIN
295 VAFIN APPR NN
296 VVINF VMINF
297 VVINF VAFIN
298 ADJA NN VVINF
299 VVFIN ART NN ART NN
300 VMFIN PIS
301 APPR ART ADJA NN VAFIN
302 PDAT NN VAFIN
303 APPR ART NN VVPP VAFIN
304 APPRART ADJA NN VVFIN
305 APPR NE VVPP
306 APPR ART NN PTKZU VVINF
307 VAFIN PAV
308 ART NN ART NN VAFIN
309 VVFIN PPOSAT NN
310 APPR ART ADJA NN VVPP
311 APPRART NN VVINF
312 ART ADJA NN VMFIN
313 VAFIN PIS
314 ADJA NN VMFIN
315 ART NN VVINF VMFIN

316 VVFIN APPR ART ADJA NN
317 VAFIN ADJA NN
318 ART NN VVPP VAINF
319 ART NN VVIZU
320 PTKZU VVINF VAFIN
321 VAFIN ART NN ART NN
322 VAFIN PDS
323 ART NN APPR ART NN VVFIN
324 VVPP VAPP VAFIN
325 VAFIN APPR ART ADJA NN
326 PAV VMFIN
327 APPR ADJA NN VVPP
328 PPOSAT NN VAFIN
329 VVFIN PPER APPR ART NN
330 VVFIN ART NN APPR ART NN
331 VVFIN APPR NE
332 ART NE VVFIN
333 ADJA NN PTKZU VVINF
334 VMFIN ART ADJA NN
335 ADJA NN VVPP VAFIN
336 PIS VAFIN
337 APPR ART NN VMFIN
338 PPOSAT NN VVINF
339 VVINF VVINF
340 APPR ART NN ART NN VVFIN
341 VVFIN APPR ADJA NN
342 APPR NN VVPP VAFIN
343 PPOSAT NN VVPP
344 ART ADJA NN PTKZU VVINF
345 VAFIN PPOSAT NN
346 ART NN ART NN VVPP
347 PIDAT NN VVFIN
348 APPR PDAT NN VVFIN
349 PAV VVPP
350 APPR ADJA NN VAFIN
351 PIAT NN VVFIN
352 ART NN APPRART NN VVFIN
353 VVINF PTKZU VMINF
354 ART ADJA NN VVPP VAFIN
355 APPRART ADJA NN VAFIN
356 PIS VMFIN
357 APPR NN PTKZU VVINF
358 APPRART NN VVPP VAFIN
359 APPR ART ADJA NN VVINF
360 VVFIN APPRART ADJA NN
361 VAFIN ART NN APPR ART NN
362 VVFIN PDS
363 VVFIN ART NN APPRART NN
364 VAFIN APPR ADJA NN
365 VVINF VVFIN
366 PDS VMFIN
367 APPR PPOSAT NN VVFIN
368 VAINF VMFIN
369 PTKZU VAINF
370 VVFIN ART NE
371 PAV VVINF
372 PPER VVINF
373 APPRART NN PTKZU VVINF
374 APPR NE VVINF
375 VMFIN APPR ART NN
376 VVPP PTKZU VAINF

377 VAFIN APPR NE
378 VAFIN PIAT NN
379 APPR PDAT NN VAFIN
380 PIS VVINF
381 PIDAT NN VAFIN
382 VVFIN PIAT NN
383 ART NN NE VVFIN
384 ART NN APPR ART NN VVPP
385 VVFIN ART NN APPR NN
386 ART NN APPR NE VVFIN
387 APPR ART NN VVPP VAINF
388 VVFIN PRF APPR ART NN
389 VVFIN ART NN NE
390 PPER VVPP
391 ART NE VAFIN
392 APPRART NN VMFIN
393 ART NN ART ADJA NN VVFIN
394 APPR ART NN VVINF VMFIN
395 APPRART NN ART NN VVFIN
396 VVFIN PPER APPRART NN
397 APPR ADJA NN VVINF
398 VAFIN APPRART ADJA NN
399 APPR ART NN ART NN VVPP
400 APPR NN VMFIN
401 VVFIN APPR PPOSAT NN
402 APPR NN KON NN VVFIN
403 ART NN APPR NN VVFIN
404 VVFIN PIDAT NN
405 VMFIN PAV
406 ART NN VAFIN VVPP
407 APPR NN VVPP VAINF
408 VMFIN VVINF
409 VMFIN APPR NN
410 PIAT NN VVINF
411 VAPP VAFIN
412 VAFIN ART NN APPR NN
413 ART NN ART NN VVINF
414 PPOSAT ADJA NN VVFIN
415 APPR ART NN VVIZU
416 ADJA NN VVPP VAINF
417 PRF VVINF
418 PDAT NN VMFIN
419 PIAT NN VAFIN
420 VMFIN APPRART NN
421 ART NN VVPP VAPP
422 APPR NE VVPP VAFIN
423 APPR ART ADJA NN VMFIN
424 APPR ART ADJA NN VVPP VAFIN
425 PIS VVPP
426 ART NN APPR ART NN VAFIN
427 ART ADJA NN VVINF VMFIN
428 ART ADJA NN ART NN VVFIN
429 ART ADJA NN VVIZU
430 VVFIN ART NN ART ADJA NN
431 PTKZU VVINF VVFIN
432 PWS VVFIN
433 APPR PPER VVFIN
434 APPRART ADJA NN VVPP
435 VVINF PTKZU VVINF
436 ART NN VVPP VAINF VMFIN
437 VAFIN PDAT NN

438 ART NN APPR ART NN VVINF
439 ART NN APPR NE VAFIN
440 ART ADJA NN VVPP VAINF
441 PRF VVFIN
442 APPR ADJA NN VVPP VAFIN
443 VVFIN PRF APPRART NN
444 PRELS VVFIN
445 PIAT NN VVPP
446 APPR PPOSAT NN VVPP
447 VAFIN ART NN APPRART NN
448 APPR NN VVINF VMFIN
449 VMFIN VVPP VAINF
450 VAFIN PPER APPR ART NN
451 VVFIN PDAT NN
452 ADJA NN VVIZU
453 PPOSAT NN PTKZU VVINF
454 ADJA NN VVINF VMFIN
455 VVFIN APPR ART NN ART NN
456 APPR ART NN ART NN VAFIN
457 VAFIN ART NE
458 APPR NN ART NN VVFIN
459 ART NN APPRART NN VAFIN
460 ART NN APPR ART ADJA NN VVFIN
461 APPR NE VMFIN
462 VAFIN APPR PDAT NN
463 ART NN NE VAFIN
464 ART NN APPR NN VVPP
465 APPR NE NE VVFIN
466 PPOSAT NN VVPP VAFIN
467 ADV ADJA NN VVFIN
468 VVFIN ART NN APPR NE
469 VVFIN APPR PDAT NN
470 APPRART NN VVPP VAINF
471 APPR ART NE VVFIN
472 APPR ART ADJA NN PTKZU VVINF
473 VVFIN PPER APPR NN
474 VVFIN ART NN NE NE
475 VMFIN PPOSAT NN
476 PWS VAFIN
477 APPR NN KON NN VVPP
478 VAFIN ART NN ART ADJA NN
479 VMFIN ADJA NN
480 APPR PPOSAT NN VVINF
481 VVFIN ART NN APPR ART ADJA NN
482 VMFIN APPR ART ADJA NN
483 VAFIN PIDAT NN
484 VAFIN APPR PPOSAT NN
485 ART NN ART ADJA NN VAFIN
486 VVFIN PRF APPR NN
487 VVFIN PRF PAV
488 ART NN APPRART NN VVPP
489 APPR ADJA NN PTKZU VVINF
490 VAFIN APPR ART NN ART NN
491 VAFIN ADV ADJA NN
492 PWAV VVFIN
493 VVFIN PPER PAV
494 APPR ART NN ART NN VVINF
495 VAFIN ART NN APPR NE
496 PIS PTKZU VVINF
497 ART PIDAT NN VVFIN
498 PRF VVPP

```

499 ART NN ART NN VMFIN
500 ART NN ART NN PTKZU VVINP
501 APPR NN VVIZU
502 APPR NN KON NN VAFIN
503 APPR PIDAT NN VVFIN
504 PIDAT NN VVPP
505 ART NN APPR NN VAFIN
506 APPR ADJA NN VMFIN
507 ART ADJA ADJA NN VVFIN
508 APPR PPOSAT NN VAFIN
509 VVFIN ADV ADJA NN
510 APPRART NN ART NN VVPP
511 VAFIN APPR PIS
512 APPR PPER VAFIN
513 VVFIN PPER APPR ART ADJA NN
514 VMFIN ART NN ART NN
515 VVFIN ART ADJA NN APPR ART NN
516 ART ADJA NN APPR ART NN VVFIN
517 APPR NE PTKZU VVINP
518 APPR ART NN VVPP VAPP
519 PDAT NN VVPP
520 APPR NN ART NN VAFIN
521 APPRART NN VVINP VMFIN
522 VVFIN APPR PIS
523 VAFIN ART NN APPR ART ADJA NN
524 VAFIN APPR NN ART NN
525 ART NN ART NN VVPP VAFIN
526 VMFIN ART NN APPR ART NN
527 PPER VAFIN VVPP
528 APPR ART ADJA NN VVPP VAINP
529 VAFIN ART ADJA NN ART NN
530 APPR ART NN ART ADJA NN VVFIN
531 # Corpus: tut-fr (Auszug aus: Frankfurter Rundschau (Juli - März ))
532 # Size: intervals/matches
533 # Size: intervals/matches
534 # Size: intervals/matches
535 # Size: intervals/matches
536 # Size: intervals/matches
537 # Size: intervals/matches
538 # Size: intervals/matches
539 # Size: intervals/matches
540 # Date: Wed Jul ::
541 # Date: Wed Jul ::
542 # Date: Wed Jul ::
543 # Date: Wed Jul ::
544 # Date: Wed Jul ::
545 # Date: Wed Jul ::
546 # Date: Wed Jul ::
547 # Date: Wed Jul ::

```

N.1.5 tags_en.txt

Englische Tagfolgen

```

1 #####NP:
2 NN
3 DT NN
4 JJ NN
5 DT JJ NN
6 NNS
7 DT NNS

```

8 NP
9 DT NP
10 NN NN
11 DT NN NN
12 NP NP
13 DT NP NP
14 JJ NNS
15 DT JJ NNS
16 JJ NN NN
17 DT JJ NN NN
18 NP NN
19 DT NP NN
20 NN NNS
21 DT NN NNS
22 NP NP NP
23 DT NP NP NP
24 JJ JJ NN
25 DT JJ JJ NN
26 DT CD NNS
27 CD NNS
28 NN NN NN
29 DT NN NN NN
30 JJ NP
31 DT JJ NP
32 JJ NP NP
33 DT JJ NP NP
34 JJ JJ NNS
35 DT JJ JJ NNS
36 JJ NN NNS
37 DT JJ NN NNS
38 NP NN NN
39 DT NP NN NN
40 NP NNS
41 DT NP NNS
42 NP NPS
43 DT NP NPS
44 NPS
45 DT NPS
46 JJ CD NNS
47 DT JJ CD NNS
48 DT CD NN
49 CD NN
50 JJ JJ NN NN
51 DT JJ JJ NN NN
52 JJR NN
53 DT JJR NN
54 NP NP NN
55 DT NP NP NN
56 NNS NN
57 DT NNS NN
58 JJ NP NN
59 DT JJ NP NN
60 JJ NN NN NN
61 DT JJ NN NN NN
62 JJS NN
63 DT JJS NN
64 DT CD NN NN
65 CD NN NN
66 NN NP
67 DT NN NP
68 NN NP NP


```

69         DT NN NP NP
70         JJ NN NP
71         DT JJ NN NP
72         RB JJ NN
73         NP NP NP NP
74         NN NN NNS
75         DT RB JJ NN
76         DT NP NP NP NP
77         DT NN NN NNS
78         NN NN NP
79         DT NN NN NP
80         DT CD JJ NNS
81         CD JJ NNS
82         JJ NP NP NP
83         DT JJ NP NP NP
84         JJS NNS
85         DT JJS NNS
86     ###***PP:
87         IN DT NN
88         IN NP
89         IN NN
90         IN DT JJ NN
91         IN DT NP
92         IN JJ NN
93         IN NNS
94         IN DT NN NN
95         IN JJ NNS
96         IN DT NNS
97         IN NP NP
98         IN NN NN
99         TO DT NN
100        IN DT NP NP
101        IN CD NN
102        IN NN NNS
103        IN DT NP NN
104        TO NP
105        TO NN
106        IN CD NNS
107        IN DT JJ NN NN
108        IN DT JJ NNS
109        IN NP NN
110        IN DT JJ JJ NN
111        IN JJ NN NN
112        TO DT NP
113        TO DT JJ NN
114        IN DT JJ NP
115        IN NP NP NP
116        IN DT NP NP NP
117        TO JJ NN
118        TO CD NN
119        IN JJ JJ NN
120        IN DT CD NN
121        IN JJ NN NNS
122        IN DT NN NNS
123        TO NNS
124        IN JJ NP
125        TO NP NP
126        IN DT NN NN NN
127        IN CD CD NN
128        TO DT NN NN
129        TO DT NNS

```

130 IN NP NNS
131 TO JJ NNS
132 IN JJ JJ NNS
133 IN JJR NN
134 IN DT NP NN NN
135 TO CD NNS
136 IN DT CD NNS
137 IN NN NN NN
138 TO DT NP NP
139 TO NN NN
140 IN DT JJ NP NP
141 IN DT JJ JJ NNS
142 IN DT JJ CD NN
143 IN CD CD NNS
144 IN DT NP NNS
145 IN DT JJ CD NNS
146 IN DT NPS
147 IN DT NP NPS
148 IN CD JJ NN
149 IN NPS
150 IN DT JJ NN NNS
151 IN NP NP NP NP
152 IN NN NP
153 IN DT JJ NP NN
154 IN RB CD NN
155 IN CD NP
156 IN DT NN NP
157 IN CD JJ NNS
158 IN NN NN NNS
159 TO DT NP NN
160 TO NN NNS
161 IN DT JJ JJ NN NN
162 IN JJR NNS
163 IN NP NPS
164 IN DT JJ NN NN NN
165 IN DT NP NP NN
166 IN DT CD NN NN
167 IN NP NN NN
168 TO DT JJ NN NN
169 IN DT NNS NN
170 IN DT JJR NN
171 IN JJ NP NP
172 IN DT JJS NN
173 IN NNS NN
174 IN NP NP NN
175 IN JJ CC JJ NN
176 IN DT JJ NN NP
177 IN CD NN NN
178 IN JJ NP NN
179 TO NP NP NP
180 TO JJ NN NN
181 TO DT JJ NNS
182 IN RB CD NNS
183 IN RB JJ NN
184 TO NP NN
185 IN JJ JJ NN NN
186 IN DT CD JJ NN
187 IN DT NP NP NP NP
188 IN DT NN NN NP
189 TO CD CD NN
190 IN CD CC CD NN

```

191 ###***VC:
192     VBD VBN
193     VB VBN
194     VBZ VBN
195     VBP VBN
196     VBZ TO VB
197     VBD TO VB
198     VBD RB VBN
199     VBZ VBG
200     VBP TO VB
201     VB TO VB
202     VBD VBN VBN
203     VBP VBG
204     VBZ RB VBN
205     VBZ VBN VBN
206     VBD VBG
207     VBD RB VB
208     VBG VBN
209     VB VBG
210     VBP VBN VBN
211     VBP RB VBN
212     VBZ RB VB
213     VB JJ TO VB
214     VBN VBN
215     VBG TO VB
216     VBZ TO VB VBN
217     VBD TO VB VBN
218     VBN TO VB
219     VBP RB VB
220     VBP TO VB VBN
221     VBD JJ TO VB
222     VBZ RB VBG
223     VBP RB VBG
224 ###***VP:
225     VBD VBN
226     VB VBN
227     VBZ VBN
228     VBP VBN
229     VBN VBN
230     VBZ TO VB
231     VBD RB VB
232     VBN TO VB
233     VBD TO VB
234     VBZ RB VB
235     VBD RB VBN
236     VBP TO VB
237     VB TO VB
238     VBZ VBG
239     VBP RB VB
240     VBZ RB VBN
241     VBP VBG
242     VBG VBN
243     VBD VBN VBN
244     VBD VBG
245     VBZ VBN VBN
246     VBG TO VB
247     VBZ VBN DT NN
248     VBP RB VBN
249     VBD VBN DT NN
250     VB VBG
251     VBP VBN VBN

```

```

252     VBD VBN NN
253     VB JJ TO VB
254     VBZ TO VB DT NN
255     VBZ TO VB VBN
256     VBN TO VB DT NN
257     VBD TO VB VBN
258     VBZ TO VB NN
259     VBZ VBN DT JJ NN
260     VBD TO VB DT NN
261     VBZ VBN NN
262     VBZ RB VBG
263     VBP VBN DT NN
264     VBD JJ TO VB
265     VBN TO VB NN
266     VBP TO VB VBN
267     VBP RB VBG
268     VBP TO VB DT NN
269     VB TO VB DT NN
270     VBD RB VB DT NN
271     VBD TO VB NN
272     VBD VBN DT JJ NN
273     VBD VBN JJ NN
274     VB VBN DT NN
275     VBP VBN NN
276     VBZ TO VB DT JJ NN
277     VBP TO VB NN
278     VB VBN NN
279     VBZ VBN JJ NN
280     VB TO VB NN
281     VBZ VBG NN
282     VBZ VBG DT NN
283     VB VBG NN
284     VBD VBN NNS
285     VBZ TO VB JJ NN
286     VBZ VBN NNS

```

N.1.6 phrasealign.py

Benutzt zuordnung2.py, um Kollokationskandidaten zu finden, und trägt diese in die Datenbank ein.

```

1  #!/usr/bin/python
2
3  #TODO:
4  #uebriggebliebene zuordnen
5  #Tabellen
6  #eintragen mit belegen
7  import sys, string
8  from DatabaseAPI import db, segmentdb, config, regiondb, qclient
9  from Region import atom, satz
10 from Phrase import zuordnung2
11 import filterwb, abstand
12
13 class PhraseAlign:
14
15     def __init__(self, db, file, opt_enterdb = 1):
16         self.opt_enterdb = opt_enterdb
17         self.db = db
18         regiondb.setDefaultDbconn(db)
19         self.segments = file.readlines()
20         self.satzidx = self.db.getMaxIndex("satzid", "saetze")
21         self.segmentidx = self.db.getMaxIndex("segmentnr", "map_s_w")

```

```

22 self.mapidx = self.db.getMaxIndex("mapid", "map_s_w")
23 self.pe_idx = self.db.getMaxIndex("pe_id", "phrase_examples")
24 self.pc_idx = self.db.getMaxIndex("pc_id", "phrase_count")
25 self.zuord = zuordnung2.OrdneZu("en", "de")
26 self.dbq = qclient.Query()
27 if not self.dbq.ok():
28     sys.stderr.write("Query-Server nicht gefunden!\n")
29 self.dbq.seterkkorpus()
30 file.close()
31
32 def align(self):
33     for segment in self.segments:
34         segment = string.strip(segment)
35         list = self.db.executelist("select segmentnr from phrase_examples where segmentnr = %s" %segment)
36         if list:
37             sys.stderr.write("segment %s already aligned!\n" %segment)
38             continue
39         phrase_de = segmentdb.SegmentVonDb(segment, self.db,
40                                           config.lang2id["de"])
41         phrase_en = segmentdb.SegmentVonDb(segment, self.db,
42                                           config.lang2id["en"])
43         zuords = self.zuord.ordne_zu(phrase_en, phrase_de)
44         print phrase_de.getText()
45         print phrase_en.getText()
46         print 36 * "*-"
47         for (p_en, p_de) in zuords:
48             self.print_segs(p_en, p_de)
49             if self.opt_enterdb:
50                 self.enter(p_de, p_en, segment)
51         print self.zuord.getInfo()
52
53 def print_segs(self, phrasel, phrase2):
54     print phrasel.getWoerter(), phrasel.getTags()
55     print phrase2.getWoerter(), phrase2.getTags()
56     print 72 * "#"
57
58 def enter(self, phrase_de, phrase_en, belegid):
59     phrase_match_de = phrase_de.getWoerter()
60     phrase_match_en = phrase_en.getWoerter()
61     if len(phrase_match_de) > len(phrase_match_en):
62         lang = "de"
63         phrase_match = string.join(phrase_match_de)
64     else:
65         lang="en"
66         phrase_match = string.join(phrase_match_en)
67     if not phrase_match:
68         return
69     #print "\tabgefragte Woerter:", phrase_match
70     segmente = self.dbq.segment(match_f = phrase_match, strict = "true", lang = lang)
71     #print "\tgefundene Segmente:", segmente
72     equ = abstand.AbstandEQU()
73     gefunden = 0
74     for segment in segmente:
75         if lang == "de":
76             segm_de = segment
77             segm_en = segment.alignedRegion()
78         else:
79             segm_en = segment
80             segm_de = segment.alignedRegion()
81     #print phrase_de.getTags(), segm_de.getTags()
82     #print phrase_en.getTags(), segm_en.getTags()

```

```

83     #print "de_equ", equ(phrase_de, segm_de)
84     #print "en_equ", equ(phrase_en, segm_en)
85     gefunden = (equ(phrase_de, segm_de) < 0.5) and (equ(phrase_en, segm_en) < 0.5)
86     if gefunden:
87         break
88     if gefunden:
89         #print "\tbenutztes Segment:", segment.getWoerter()
90         segid = db.unknown2string(segment.regionid)
91         list = self.db.executelist(
92             ("select pc_id, zaehler ") +\
93             ("from phrase_count ") +\
94             ("where segmentnr = %s" %(segid))
95         )
96         if list:
97             pc_idx, zaehler = list[0]
98             zaehler = self.db.incr(zaehler)
99             self.db.update(
100                ("update phrase_count ") +\
101                ("set zaehler = %s " %zaehler) +\
102                ("where pc_id = %s" %db.unknown2string(pc_idx))
103            )
104         else: #nicht gefunden
105             langs = {"de" : phrase_de, "en": phrase_en}
106             for i in langs.keys():
107                 lang_id = config.lang2id[i]
108                 self.db.update(
109                     ("insert into saetze ") +\
110                     ("(satzid, sprachenid, herkunftsids) ") +\
111                     ("values(%s, %s, %s)" %(self.satzidx, lang_id, -1))
112                 )
113                 #print "satzid:", self.satzidx
114                 #print "sprache:", lang_id
115                 #print "segment:", self.segmentidx
116                 self.enterphrase(langs[i], self.satzidx, i)
117                 self.satzidx = self.db.incr(self.satzidx)
118             self.db.update(
119                 ("insert into phrase_count ") +\
120                 ("(pc_id, segmentnr, zaehler) ") +\
121                 ("values(%s, %s, 1)" %(self.pc_idx, self.segmentidx))
122             )
123             self.pc_idx = self.db.incr(self.pc_idx)
124             #todo: das folgende schoener machen!!!
125             self.dbq.query("cache.delete t:segment;s:text;w:strict=true,lang=%s,match=%s" %(lang, phrase_matc
126             self.dbq.query("cache.delete t:segment;s:pyob;w:match_f=%s,strict=%s,lang=%s" %(phrase_match, "tr
127             self.db.update(
128                 ("insert into phrase_examples ") +\
129                 ("(pe_id, pc_id, segmentnr) ") +\
130                 ("values(%s, %s, %s)" %(self.pe_idx, self.pc_idx, belegid))
131             )
132             self.segmentidx = self.db.incr(self.segmentidx)
133             self.pe_idx = self.db.incr(self.pe_idx)
134
135     def enterphrase(self, phrase, satzid, lang):
136         all = map(None, phrase.getWoerter(), phrase.getTags(), phrase.getLemmata())
137         for (word, tag, gf) in all:
138             # print word, tag, gf
139             tokenid = self.db.executelist(
140                 ("select t.tokenid ") +\
141                 ("from tokens t, grundformen gf, tagset_ims_%s tags " %lang) +\
142                 ("where t.name = '%s' " %word) +\
143                 ("and t.grundformid = gf.grundformid ") +\

```

```

144         ("and gf.name = '%s' " %gf) +\
145         ("and t.tagid = tags.tagid ") +\
146         ("and tags.name = '%s' " %tag)
147     )
148     print tokenid
149     if tokenid:
150         tokenid = tokenid[0][0]
151         self.db.update(
152             ("insert into map_s_w ") +\
153             ("(mapid, tokenid, satzid, segmentnr) ") +\
154             ("values(%s, %s, %s, %s)" %(self.mapidx,
155                                     db.unknown2string(tokenid),
156                                     satzid,
157                                     self.segmentidx
158                                     )))
159     )
160     self.mapidx = self.db.incr(self.mapidx)
161
162     def close(self):
163         self.db.close()
164         self.dbq.close()
165
166     def usage():
167         print "Usage: phrasealign.py [-n] [<file with segment-IDs>]"
168         print "-n          do not enter stuff into database"
169         sys.exit()
170
171     def main():
172         import getopt
173         # process command line args
174         try:
175             opts, args = getopt.getopt(sys.argv[1:], "hn", ["help", "nodb"])
176         except getopt.GetoptError:
177             # print help information and exit:
178             usage()
179         opt_enterdb = 1
180         for o, a in opts:
181             if o in ("-n", "--nodb"):
182                 opt_enterdb = 0
183             if o in ("-h", "--help"):
184                 usage()
185         dbconn = db.Db()
186         file = sys.stdin
187         if len(args) > 0:
188             file = open(args[0], "r")
189         paligner = PhraseAlign(dbconn, file, opt_enterdb)
190         paligner.align()
191         paligner.close()
192
193     if __name__ == '__main__':
194         main()
195

```

N.1.7 demo-zuord2.py

```

1  #!/usr/bin/python
2
3  from DatabaseAPI import segmentdb, db, regiondb
4  from Phrase import zuordnung2
5  import string
6  import sys

```

```

7
8 def main():
9     dbconn = db.Db()
10    regiondb.setDefaultDbconn(dbconn)
11    reg1 = segmentdb.SegmentVonDb(111219, dbconn, "1")
12    reg2 = segmentdb.SegmentVonDb(111219, dbconn, "2")
13    print reg1.getText()
14    print reg2.getText()
15    ord = zuordnung2.OrdneZu("de", "en")
16    phrases = ord.ordne_zu(reg1, reg2)
17    for (i, j) in phrases:
18        print i.getText()
19        print j.getText()
20        print 72 * "-"
21    dbconn.close()
22
23 if __name__ == '__main__':
24     main()
25
26

```

N.1.8 mogel.py

Dieses Skript hilft beim manuelle Einfügen von erkannten Phrasen mit Beispielsätzen in die Datenbank.

```

1  #!/usr/bin/python
2
3  import korpus2db
4
5  # Eingabedatei: nacheinander immer 3 Segmente: de, en, ID-Liste der Beispiele
6
7  import sys
8  import string
9
10 k = korpus2db.Korpus()
11 f = open(sys.argv[1], "rt")
12 segliste = string.split(f.read(), "<segmentgrenze>")
13 tagset_de = k.dbconn.executelist("select name, tagsetid from tagsets where sprachenid = %s and name = '%s'" %
14 tagset_en = k.dbconn.executelist("select name, tagsetid from tagsets where sprachenid = %s and name = '%s'" %
15 tagsets=[tagset_de[0], tagset_en[0]]
16 sprachen = ["de", "en"]
17 pc_id = k.dbconn.getMaxIndex("pc_id", "phrase_count")
18 pe_id = k.dbconn.getMaxIndex("pe_id", "phrase_examples")
19 for i in range(len(segliste)):
20     segment = [segliste[i]]
21     #sys.stderr.write("segment = %s\n" %`segment`)
22     sprache = 1 + (i % 3)
23     if sprache == 1:
24         segstart = k.segment_idx
25         sys.stderr.write("Processing %s -> %s\n" %(i,segstart))
26     elif sprache == 2:
27         k.segment_idx = segstart
28     elif sprache == 3:
29         idlistestr = string.split(segment[0])
30         idliste = []
31         for idstr in idlistestr:
32             if idstr:
33                 idliste.append(int(idstr))
34         k.dbconn.update("insert into phrase_count(pc_id, segmentnr, zaehler) values (%s, %s, %s)" %(pc_id, se
35         for seg_id in idliste:
36             k.dbconn.update("insert into phrase_examples(pe_id, pc_id, segmentnr) values(%s, %s, %s)" %(pe_id,

```



```

37         pe_id = k.dbconn.incr(pe_id)
38         sys.stderr.write("%s Beispiele fuer Phrase %s\n" %(len(idliste),pc_id))
39         pc_id = k.dbconn.incr(pc_id)
40         continue
41     else:
42         raise IndexError
43     k.tag_idx = k.dbconn.getMaxIndex("tagid", "tagset_ims_%s" %sprachen[sprache-1])
44     k.entersegmente(segment, k.sent_idx, -1, sprache, tagsets[sprache-1])
45 f.close()
46 k.close()

```

N.1.9 web_phrases.py

Die erkannten Phrasen und ihre Beispiele (Belege) können mit diesem CGI-Skript dargestellt werden. Bei der Installation im cgi-bin Verzeichnis ist zu beachten, dass die Pakete Region und DatabaseAPI verfügbar sein müssen.

```

1  #!/usr/bin/python
2
3  from DatabaseAPI import segmentdb, config, regiondb, db
4  from Region import atom
5  import base64
6  import cgi
7  import pickle
8  import socket
9  import string
10 import sys
11 import urllib
12
13 # web_phrases.py ist ein CGI-Skript, das die von KoKS selbst erkannten
14 # Phrasen und die zugehoerigen Beispiele tabellarisch darstellt.
15 # Joachim Wagner, 24. Nov. 2001
16
17 scriptname = urllib.quote('web_phrases.py')
18
19 print "Content-type: %s\n" %"text/html" # Extra-Zeilentrenner wichtig!
20
21 def kopf(titel):
22     print """<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
23     <html>
24     <head>
25         <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
26         <title>%s</title>
27     </head>
28     <body>
29     <h1>%s</h1>
30     "" " %(titel, titel)
31
32 def ende():
33     print "</body></html>"
34
35 def printseg(segdb):
36     segdb_str=pickle.dumps(segdb)
37     segdb_64=base64.encodestring(segdb_str)
38     print '<a href="%s?segment=%s">' %(scriptname, urllib.quote(segdb_64))
39     print cgi.escape(segdb.getText())
40     print '</a>'
41
42 def seg_sort_fun(a,b):
43     token_a = dbconn.executelist(
44         "select tokenid " +

```

```

45     "from map_s_w " +
46     "where segmentnr = %s" %a
47 )
48 token_b = dbconn.executelist(
49     "select tokenid " +
50     "from map_s_w " +
51     "where segmentnr = %s" %b
52 )
53 return len(token_a) - len(token_b)
54
55 dbconn = db.Db()
56 regiondb.setDefaultDbconn(dbconn)
57 try:
58     form = cgi.FieldStorage()
59     form_ok = 0
60     if form.has_key("phrsid"):
61         phrsid = form["phrsid"].value
62         kopf("Beispiele zur Phrase %s" %phrsid)
63         orderby = ""
64         sortl = 1 # Beispiele nach Laenge sortieren
65         try:
66             orderby = form["sort"].value
67             sortl = 0
68             if orderby == "length":
69                 sortl = 1
70                 orderby = ""
71                 print '<p>Sortiert nach Länge.</p>'
72             if orderby not in ["segmentnr", "pe_id", "pc_id"]:
73                 print '<p>Fehler: %s kein gültiges Sortierkriterium</p>' %cgi.quote(orderby)
74                 orderby = ""
75             else:
76                 print '<p>Sortiert nach %s.</p>' %orderby
77                 orderby = " order by " + orderby
78         except:
79             pass
80         data = dbconn.executelist(
81             "select pe_id, segmentnr " +
82             "from phrase_examples " +
83             ("where pc_id = %s" %phrsid) + orderby
84         )
85         print '<table border="1">'
86         print '<tr>'
87         print '<th><a href="%s?phrsid=%s&sort=pe_id">pe_id</a></th>' %(scriptname,urllib.quote(phrsid))
88         print '<th><a href="%s?phrsid=%s&sort=segmentnr">seg_id</a></th>' %(scriptname,urllib.quote(phrsid))
89         print '<th>Deutsch</th><th>Englisch</th>'
90         print '</tr>'
91         if sortl:
92             data.sort(lambda x,y: seg_sort_fun(int(x[1]),int(y[1])))
93         for (pe_id, segmentnr) in data:
94             print '<tr><td align="right">'
95             print int(pe_id)
96             print '</td><td align="right">'
97             print int(segmentnr)
98             print '</td><td>'
99             printseg(segmentdb.SegmentVonDb(int(segmentnr), dbconn, "1"))
100            print '</td><td>'
101            printseg(segmentdb.SegmentVonDb(int(segmentnr), dbconn, "2"))
102            print '</td></tr>'
103        print '</table>'
104        if not sortl:
105            print '<p><a href="%s?phrsid=%s&sort=length">Nach Länge sortieren.</a></p>' %(scriptname,urllib.q

```

```

106     print '<p><a href="%s">Zurück zur Phrasenübersicht</a></p>' %scriptname
107 elif form.has_key("segment"):
108     segdb_64 = form["segment"].value
109     segdb_str = base64.decodestring(segdb_64)
110     segdb = pickle.loads(segdb_str)
111     kopf("Details zu %s" %cgi.escape(segdb.getText()))
112     print '<table border="1">'
113     print '<tr><th>Wort</th><th>Tag</th><th>Lemma</th></tr>'
114     wtl = map(None, segdb.getWoerter(),
115               segdb.getTags(),
116               segdb.getLemmata())
117     for line in wtl:
118         print '<tr>'
119         for item in line:
120             print '<td>%s</td>' %cgi.escape(item)
121         print '</tr>'
122     print '</table>'
123     print '<p><a href="%s">Zurück zur Phrasenübersicht</a></p>' %scriptname
124 else:
125     kopf("Phrasen mit mindestens zwei Beispielen")
126     print "<p>Einzelworteinträge ausgelassen.</p>"
127     data = dbconn.executelist(
128         "select pc_id, segmentnr, zaehler "
129         "from phrase_count "
130         "where zaehler > 1 "
131         "order by zaehler desc"
132     )
133     print '<table border="1">'
134     print '<tr><th>pc_id</th><th>Beispiele</th><th>Deutsch</th><th>Englisch</th></tr>'
135     for (pc_id, segmentnr, zaehler) in data:
136         segde = segmentdb.SegmentVonDb(int(segmentnr), dbconn, "1")
137         segen = segmentdb.SegmentVonDb(int(segmentnr), dbconn, "2")
138         atmde = segde.getRegionenMit(atom.Atom)
139         atmen = segen.getRegionenMit(atom.Atom)
140         if len(atmde) == 1 and len(atmen) == 1:
141             continue
142         print '<tr><td>'
143         print int(pc_id)
144         print '</td><td>'
145         print '<a href="web_phrases.py?phrsid=%s">%s</a>' %(int(pc_id),int(zaehler))
146         print '</td><td>'
147         printseg(segde)
148         print '</td><td>'
149         printseg(segen)
150         print '</td></tr>'
151     print '</table>'
152     ende()
153     dbconn.close()
154 except socket.error, data:
155     kopf("Fehler")
156     print socket.error, data
157     ende()
158     dbconn.close()

```

Anhang O

Datenbank – Code

O.1 SQL-Skripte zum Erstellen der Datenbank

O.1.1 db.sql

Diese Skript erzeugt die Datenbank.

```
1 CREATE TABLE autoren (  
2     autorid int8 primary key,  
3     name varchar(100) BINARY);  
4 CREATE TABLE herkunft (  
5     herkunftsaid int8 primary key,  
6     name varchar(100) BINARY,  
7     autorid int8 references autoren,  
8     jahr varchar(30) BINARY,  
9     quellenid int8 references quellen);  
10 CREATE TABLE quellen (  
11     quellenid int8 primary key,  
12     name varchar(100) BINARY,  
13     verlag varchar(100) BINARY,  
14     rechte varchar(200) BINARY,  
15     bemerkungen varchar(200) BINARY,  
16     statusid int4);  
17 CREATE TABLE tokens (  
18     tokenid int8 primary key,  
19     name varchar(50) BINARY not null,  
20     reverse varchar(50) BINARY not null,  
21     grundformid int8 not null references grundformen on delete cascade,  
22     tagid int8 not null,  
23     tagqualitaet int4,  
24     tagsetid int8);  
25 CREATE TABLE saetze (  
26     satzid int8 primary key,  
27     herkunftsaid int8 not null references herkunft on delete cascade,  
28     parsequalitaet int4,  
29     segmentqual int4,  
30     prev int8,  
31     succ int8,  
32     sprachenid int8 not null);  
33 CREATE TABLE map_s_w (  
34     mapid int8 primary key,  
35     satzid int8 not null references saetze on delete cascade,  
36     tokenid int8 not null references tokens,  
37     segmentnr int8 not null);  
38 CREATE TABLE grundformen (  
39     grundformid int8 primary key,  
40     name varchar(100) BINARY not null,
```

```

41     reverse varchar(100) BINARY not null,
42     tagid int8 not null,
43     tagsetid int8 not null);
44 CREATE TABLE phrase_count(
45     pc_id int8 primary key,
46     segmentnr int8 not null references map_s_w on delete cascade,
47     zaehler int8);
48 CREATE TABLE phrase_examples(
49     pe_id int8 primary key,
50     pc_id int8 references phrase_count on delete cascade,
51     segmentnr int8 not null references map_s_w on delete cascade);
52 CREATE TABLE sprachen (
53     sprachenid int8 primary key,
54     name varchar(100) BINARY);
55 CREATE TABLE tagsets (
56     tagsetid int8 primary key,
57     name varchar(100) BINARY,
58     sprachenid int8);
59 CREATE TABLE tagset_ims_de (
60     tagid int8 primary key,
61     name varchar(100) BINARY);
62 CREATE TABLE tagset_ims_en (
63     tagid int8 primary key,
64     name varchar(100) BINARY);
65 CREATE TABLE tagset_lql_de (
66     tagid int8 primary key,
67     name varchar(100) BINARY);
68 CREATE TABLE tagset_lql_en (
69     tagid int8 primary key,
70     name varchar(100) BINARY);
71 CREATE TABLE status (
72     statusid int8 primary key,
73     name varchar(100) BINARY);
74 insert into tagsets (tagsetid, name, sprachenid) values (1, 'tagset_ims_de', 1);
75 insert into tagsets (tagsetid, name, sprachenid) values (2, 'tagset_ims_en', 2);
76 insert into tagsets (tagsetid, name, sprachenid) values (3, 'tagset_lql_de', 1);
77 insert into tagsets (tagsetid, name, sprachenid) values (4, 'tagset_lql_en', 2);
78 insert into herkunft(herkunftsid, name, autorid, jahr, quellenid) values (-1, "erkannt", -1, null, -1);
79 create index grundformid_idx on tokens (grundformid);
80 create index tagid_idx on tokens (tagid);
81 create index segmentnr_index on map_s_w(segmentnr);
82 create index satzid_index on map_s_w(satzid);
83 create index tokenid_index on map_s_w(tokenid);
84 create index sprachenid_index on saetze(sprachenid);
85 create index herkunftsid_index on saetze(herkunftsid);
86 create index tokenname_index on tokens (name);
87 create index reverse_index on tokens (reverse);
88 create index gf_reverse_index on grundformen (reverse);
89 create index grundformname_index on grundformen (name);
90 create index pe_id_index on phrase_examples(pe_id);
91 create index pc_segment_index on phrase_count(segmentnr);
92 create index pc_id_index on phrase_count(pc_id);
93 create index pe_segment_index on phrase_examples(segmentnr);

```

O.1.2 index.sql

Dieses Skript fügt die Indizes hinzu.

```

1 create unique index tokenid_idx on tokens (tokenid);
2 create unique index tag_de_idx on tagset_ims_de (tagid);
3 create unique index tag_en_idx on tagset_ims_en (tagid);

```

```

4 create unique index autorid_idx on autoren (autorid);
5 create unique index herkunftsidx on herkunft (herkunftsidx);
6 create unique index quellenid_idx on quellen (quellenid);
7 create unique index mapid_idx on map_s_w (mapid);
8 create unique index sprachenid_idx on sprachen (sprachenid);
9 create unique index statusid_idx on status (statusid);
10 create unique index tagsetid_idx on tagsets (tagsetid);
11 create unique index satzid_idx on saetze (satzid);
12 create unique index grundformid_index on grundformen (grundformid);
13 /*create index grundformid_idx on tokens (grundformid);
14 create index tagid_idx on tokens (tagid);
15 create index segmentnr_index on map_s_w(segmentnr);
16 create index satzid_index on map_s_w(satzid);
17 create index tokenid_index on map_s_w(tokenid);
18 create index sprachenid_index on saetze(sprachenid);
19 create index herkunftsidx_index on saetze(herkunftsidx);
20 create index grundformname_index on grundformen (name);
21 create index tokenname_index on tokens (name);*/
22 /*create unique index gf_name_tag_index on grundformen (name, tagid);
23 create unique index tk_name_tag_gf_index on tokens(name, tagid, grundformid);*/

```

O.1.3 clean.sql

Mit diesem Skript löscht man aus allen Tabellen, in die beim Eintragen von Korpora geschrieben wird.

```

1 DELETE from tokens;
2 DELETE from saetze;
3 DELETE from map_s_w;
4 DELETE from autoren;
5 DELETE from quellen;
6 DELETE from herkunft;
7 DELETE from grundformen;
8 DELETE from tagset_ims_de;
9 DELETE from tagset_ims_en;

```

O.1.4 mrproper.sql

Dieses Skript löscht alle angelegten Tabellen.

```

1 drop table autoren;
2 drop table grundformen;
3 drop table herkunft;
4 drop table map_s_w;
5 drop table phrase_count;
6 drop table phrase_examples;
7 drop table quellen;
8 drop table saetze;
9 drop table sprachen;
10 drop table status;
11 drop table tagset_ims_de;
12 drop table tagset_ims_en;
13 drop table tagset_lql_de;
14 drop table tagset_lql_en;
15 drop table tagsets;
16 drop table tokens;

```

O.2 Python-Datenbank-Schnittstelle

Das Python-Paket DatabaseAPI befindet sich im CVS Baum im Verzeichnis align.

O.2.1 `__init__.py`

```

1 #!/usr/bin/python
2
3 # Initialisierung des Pakets "DatabaseAPI"
4
5 __all__ = [                                # Module als DatabaseAPI.xxx ansprechbar machen
6     "absatzdb.py",
7     "atomdb.py",
8     "cache.py",
9     "config.py"
10    "db.py",
11    "dokumentdb.py",
12    "korpusdb.py",
13    "qclient.py",
14    "querydb.py",
15    "regiondb.py",
16    "satzdb.py",
17    "segmentdb.py",
18 ]
19
20 # nicht oeffentlich:
21 #     "alignmentdb.py",
22 #     "hindex.py",
23 #     "secret.py",

```

O.2.2 RegionVonDB

`regiondb.py`

Die Grundklasse aller Datenbank-Regionen.

```

1 #!/usr/bin/python
2
3 # Modul DatabaseAPI.regiondb
4
5 from Region import region
6 import db
7 import sys
8 import types
9
10 defaultDbconn = None
11
12 def setDefaultDbconn(dbconn):
13     global defaultDbconn
14     defaultDbconn = dbconn
15
16 class RegionVonDb(region.Regionen):
17
18     def __init__(self, id, dbconn):
19         self.dbconn = dbconn
20         self.subregionen = {}
21         self.regionid = id
22         self.idliste = self.loadids(id)
23
24     def __repr__(self):    # ueberschreiben
25         return "<%s id=%s>" %(self.__class__.__name__, self.regionid) #, self.subregionen)
26
27     def __getstate__(self):
28         return (self.idliste, self.regionid, self.subregionen)
29
30     def __setstate__(self, state):

```

```

31     self.dbconn = defaultDbconn
32     self.idliste = state[0]
33     self.regionid = state[1]
34     #self.subregionen = {}
35     self.subregionen = state[2]
36
37     def setDbconn(self, dbconn):
38         self.dbconn = dbconn
39         for key in self.subregionen.keys():
40             self.subregionen[key].setDbconn(dbconn)
41
42     def loadids(self, id): #ueberschreiben
43         return []
44
45     def loaditem(self, key, pos): #ueberschreiben
46         return ""
47
48     def loadquery(self, query):
49         #sys.stderr.write("RegionVonDb.loadquery("+`query`+")\n")
50         ids = []
51         result = self.dbconn.executelist(query)
52         if result and result[0]:
53             ids = map(lambda x: db.unknown2string(x[0]), result)
54         return ids
55
56     def __len__(self):
57         #if type(self.idliste) != types.ListType:
58         #    sys.stderr.write("Fehler: type(RegionVonDb.idliste) = "+`type(self.idliste)`+"\n")
59         return len(self.idliste)
60
61     def getItemAtIndex(self, key):
62         if not self.subregionen.has_key(key):
63             self.subregionen[key] = self.loaditem(self.idliste[key])
64         return self.subregionen[key]
65
66

```

alignmentdb.py

Repräsentation einer alignnten Region.

```

1  #!/usr/bin/python
2
3  # Modul DatabaseAPI.alignmentdb
4
5  import regiondb, db
6  from Region import alignment
7  import config
8  import sys
9  import types
10
11 class AlignedRegionVonDb(regiondb.RegionVonDb, alignment.AlignedRegion): #{
12
13     def __init__(self, id, dbconn, sprache):
14         self.sprache = sprache
15         self.alregid = id
16         regiondb.RegionVonDb.__init__(self, id, dbconn)
17
18     def __repr__(self):
19         return "<"+db.unknown2string(self.alregid)+" "+\
20             self.__class__.__name__+" "+\
21             db.unknown2string(self.sprache)+" "+\

```



```

22     ("dbconn=%s" % 'id(self.dbconn)')+ ">"
23
24     def alignedRegion(self):
25         sprache = None
26         for i in config.lang2id.values():
27             if i != self.sprache:
28                 sprache = i
29         if sprache:
30             return self.__class__(self.alregid, self.dbconn, sprache)
31         else:
32             sys.stderr.write("AlignedRegionVonDb: alignte Sprache unklar.\n")
33
34     def __setstate__(self, state):
35         regiondbstate, sprache, alregid = state
36         regiondb.RegionVonDb.__setstate__(self, regiondbstate)
37         self.sprache = sprache
38         self.alregid = alregid
39
40     def __getstate__(self):
41         regiondbstate = regiondb.RegionVonDb.__getstate__(self)
42         return (regiondbstate, self.sprache, self.alregid)
43
44 #}

```

atomdb.py

Repräsentation eines Wortes.

```

1  #!/usr/bin/python
2
3  # Modul DatabaseAPI.atomdb
4
5  from Region import atom
6  import db, regiondb
7  import sys
8
9  #Problem: nimmt immer nur daserste Token!!!! TODO
10 class AtomVonDb(atom.Atom):
11
12     def __repr__(self):
13         return "<AtomVonDB " + '( self.feld["Wort"],
14                               self.feld["Tag"],
15                               self.feld["Lemma"] )' + ">"
16
17     def __init__(self, wort, dbconn, id = 0):
18         # atom.Atom.__init__(self) # Aufruf erfolgt in loadFromDb
19         self.dbconn = dbconn
20         self.loadFromDb(wort, id)
21
22     def __getstate__(self):
23         return self.id
24
25     def __setstate__(self, state):
26         self.dbconn = regiondb.defaultDbconn
27         self.loadFromDb("", state)
28
29     def setDbconn(self, dbconn):
30         self.dbconn = dbconn
31
32     def loadFromDb(self, wort, id): # id ist String oder Int, aber kein Long
33         tag = "<none>"
34         lemma = "<unknown>"

```

```

35     if id:
36         where = "tokens.tokenid = %s" %id
37     else:
38         where = "tokens.name = '%s'" %wort
39     results = self.dbconn.executelist("select distinct grundform.name, tokens.tagid, tokens.tagsetid, tokens.tokenid
40     if results:
41         lemma = results[0][0]
42         tagid = db.unknown2string(results[0][1])      # dieses spaeter???
43         tagsetid = db.unknown2string(results[0][2])
44         wort = results[0][3]
45         self.id = db.unknown2string(results[0][4])
46         results = self.dbconn.executelist("select name from tagsets where tagsetid = %s" %tagsetid)
47         if results:
48             tagset = db.unknown2string(results[0][0])
49             results = self.dbconn.executelist("select name from %s where tagid = %s" %(tagset, tagid))
50             tag = db.unknown2string(results[0][0])
51         atom.Atom.__init__(self, wort, tag, lemma)
52
53 def test():
54     dbconn = db.Db()
55     for i in ("hat", "is", "pension", "Steuer", "er", "and"):
56         atom = AtomVonDb(i, dbconn)
57         print i, atom, 'atom', atom.getWoerter()
58     for i in (3, 76, 129):
59         atom = AtomVonDb("", dbconn, i)
60         print i, atom, 'atom', atom.getWoerter()
61
62 if __name__ == '__main__':
63     test()

```

satzdb.py

Repräsentation eines Satzes.

```

1  #!/usr/bin/python
2
3  # Modul DatabaseAPI.satzdb
4
5  from Region import satz
6  import regiondb, atomdb, db
7  import sys
8
9  class SatzVonDb(regiondb.RegionVonDb, satz.Satz):
10
11     def __init__(self, id, dbconn):
12         regiondb.RegionVonDb.__init__(self, id, dbconn)
13
14     #def __repr__(self):      #ueberschreiben
15     # return "<Satz von Db>" # + 'self.subregionen' + ">"
16
17     def loadids(self, id):
18         #sys.stderr.write("SatzVonDb.loadids('+id+')\n")
19         retval = self.loadquery("select tokenid from map_s_w where satzid = %s order by mapid" %id)
20         #sys.stderr.write("idliste = " + 'retval' + "\n")
21         return retval
22
23     def loaditem(self, key):
24         return atomdb.AtomVonDb("", self.dbconn, key)
25
26 def test():
27     dbconn = db.Db()
28     for i in (1, '23', 40, '190000', 397433, '400293'):

```

```

29     satz = SatzVonDb(i, dbconn)
30     print `(i, satz, satz.getText())`
31
32 if __name__ == `__main__`:
33     test()

```

segmentdb.py

Repräsentation eines Segments.

```

1  #!/usr/bin/python
2
3  from Region import segment
4  import satzdb, db, alignmentdb
5  import sys
6
7  class SegmentVonDb(alignmentdb.AlignedRegionVonDb, segment.Segment):
8
9      def __init__(self, id, dbconn, sprache):
10         alignmentdb.AlignedRegionVonDb.__init__(self, id, dbconn, sprache)
11
12     def loadids(self, id):
13         #sys.stderr.write("SegmentVonDb.loadids: " + `(id, self.sprache)` + "\n")
14         return self.loadquery("select distinct m.satzid from map_s_w m, saetze s where m.segmentnr = %s and m.satzid = %s")
15
16     def loaditem(self, key):
17         return satzdb.SatzVonDb(key, self.dbconn)
18
19 def test():
20     dbconn = db.Db()
21     for i in (1, 2, 150000, 155100, 167700, 179888, 190000):
22         de = SegmentVonDb(i, dbconn, "1")
23         en = de.alignedRegion()
24         print `(i, de, en)`, de.getText(), en.getText()
25
26 if __name__ == `__main__`:
27     test()
28

```

absatzdb.py

Repräsentation eines Absatzes.

```

1  #!/usr/bin/python
2  from Region import absatz
3  import segmentdb, db, alignmentdb
4
5  #bis jetzt ist ein Absatz das gleiche wie ein Dokument!
6
7  class AbsatzVonDb(alignmentdb.AlignedRegionVonDb, absatz.Absatz):
8
9      def __init__(self, id, dbconn, sprache):
10         alignmentdb.AlignedRegionVonDb.__init__(self, id, dbconn, sprache)
11
12     def loadids(self, id):
13         return self.loadquery("select distinct m.segmentnr from map_s_w m, saetze s where s.herkunftsid = %s and m.satzid = %s")
14
15     def loaditem(self, key):
16         return segmentdb.SegmentVonDb(key, self.dbconn, self.sprache)
17
18 def test():
19     dbconn = db.Db()

```

```

20     for i in (10,):
21         absatz = AbsatzVonDb(i, dbconn, "1")
22         print `absatz`
23         print absatz[1][0][0]
24         print absatz[1][0]
25         print absatz[1]
26         print absatz.alignedRegion()[1]
27
28 if __name__ == '__main__':
29     test()

```

dokumentdb.py

Repräsentation eines Dokuments.

```

1  #!/usr/bin/python
2  from Region import dokument
3  import absatzdb, db, alignmentdb
4
5  #bis jetzt ist ein Absatz das gleiche wie ein Dokument!
6
7  class DokumentVonDb(alignmentdb.AlignedRegionVonDb, dokument.Dokument):
8
9      def __init__(self, id, dbconn, sprache):
10         alignmentdb.AlignedRegionVonDb.__init__(self, id, dbconn, sprache)
11
12     def loadids(self, id):
13         return [id]
14
15     def loaditem(self, key):
16         return absatzdb.AbsatzVonDb(key, self.dbconn, self.sprache)
17
18 def test():
19     dbconn = db.Db()
20     for i in (1, 2):
21         dokument = DokumentVonDb(i, dbconn, "1")
22         print `dokument`
23
24 if __name__ == '__main__':
25     test()

```

korpusdb.py

Repräsentation eines Korpus.

```

1  #!/usr/bin/python
2  from Region import korpus
3  import dokumentdb, db, alignmentdb
4
5  #bis jetzt ist ein Absatz das gleiche wie ein Dokument!
6  class KorpusVonDb(alignmentdb.AlignedRegionVonDb, korpus.KorpusMitSprache):
7
8      def __init__(self, dbconn, sprache):
9         alignmentdb.AlignedRegionVonDb.__init__(self, 0, dbconn, sprache)
10
11     def __repr__(self): #ueberschreiben
12         return "<Korpus von Db>"
13
14     def loadids(self, id):
15         ids = []
16         result = self.dbconn.executelist("select herkunftsaid from herkunft")
17         if result and result[0]:

```

```

18         ids = map(lambda x: x[0], result)
19         return ids
20
21     def loaditem(self, key):
22         return dokumentdb.DokumentVonDb(key, self.dbconn, self.sprache)
23
24
25 #Falle! Sprachids sind "1" => deutsch, "2" => englisch
26 def test():
27     dbconn = db.Db()
28     for i in ("1", "2"):
29         korpus = KorpusVonDb(dbconn, i)
30         print korpus[0][0][0][0]
31
32 if __name__ == '__main__':
33     test()

```

O.2.3 Wörterbücher

wb.py

Ein Wörterbuch, aus einer Datei oder der Datenbank.

```

1  #!/usr/bin/python
2
3  # Modul wb
4
5  from DatabaseAPI import cache
6  import zeitmessung
7  import string
8  import sys
9  import time
10 import types
11 # weitere imports nach den Klassendefinitionen
12
13 class Lemma2src(cache.Cached): #{
14
15     def __init__(self, client, quellsprache, zielsprache):
16         cache.Cached.__init__(self, client)
17         self.quellsprache = quellsprache
18         self.zielsprache = zielsprache
19
20     def loaditem(self, key):
21         return []
22
23     def saveitem(self, key, value):
24         pass
25
26 #}
27
28 class Src2dst(cache.Cached): #{
29
30     def __init__(self, client, quellsprache, zielsprache):
31         cache.Cached.__init__(self, client)
32         self.quellsprache = quellsprache
33         self.zielsprache = zielsprache
34
35     def loaditem(self, key):
36         return []
37
38     def saveitem(self, key, value):
39         pass

```

```

40
41 #}
42
43 class Woerterbuch(zeitmessung.Zeitmessung): #{
44
45     def __init__(self, quellsprache, zielsprache):
46         zeitmessung.Zeitmessung.__init__(self)
47         self.lemma2src = Lemma2src(self, quellsprache, zielsprache)
48         self.src2dst = Src2dst(self, quellsprache, zielsprache)
49         self.quellsprache = quellsprache
50         self.zielsprache = zielsprache
51
52     def __repr__(self):
53         return "<WB %s nach %s>" % (self.quellsprache, self.zielsprache)
54
55     def __getitem__(self, lemma):
56         self.p_begin("getitem")
57         srcliste = self.lemma2src[lemma]
58         retval=[]
59         for (pos, srcseg) in srcliste:
60             retval.append((pos, srcseg, self.src2dst['srcseg']))
61         self.p_end("getitem")
62         return retval
63
64     def has_key(self, lemma):
65         self.p_begin("has_key")
66         retval = self.lemma2src.has_key(lemma)
67         self.p_end("has_key")
68         return retval
69
70 #}
71
72 #####
73 #
74 # VonDatei
75
76 class Lemma2srcVonDateien(Lemma2src): #{
77
78     def __init__(self, client, quellsprache, zielsprache):
79         Lemma2src.__init__(self, client, quellsprache, zielsprache)
80         if not isinstance(self, cache.FullCache):
81             sys.stderr.write("Warnung: Lemma2srcVonDateien basiert nicht auf FullCache\n")
82
83     def addSegmentpaar(self, seg_s, seg_d):
84         pos = 1 # Bei jedem Lemma seg_s eintragen
85         for lemma in seg_s.getLemmata(): #{
86             if self.has_key(lemma):
87                 srcliste = self[lemma]
88             else:
89                 srcliste = []
90             srcliste.append((pos, seg_s)) # bei der Gelegenheit pos merken
91             self[lemma] = srcliste
92             pos = pos + 1
93         #}
94 #}
95
96 class Src2dstVonDateien(Src2dst): #{
97
98     def __init__(self, client, quellsprache, zielsprache):
99         Src2dst.__init__(self, client, quellsprache, zielsprache)
100         if not isinstance(self, cache.FullCache):

```

```

101     sys.stderr.write("Warnung: Src2dstVonDateien basiert nicht auf FullCache\n")
102
103 def addSegmentpaar(self, seg_s, seg_d):
104     srckey = 'seg_s' # Paar (seg_s, seg_d) eintragen
105     if self.has_key(srckey):
106         dstliste = self[srckey]
107     else:
108         dstliste = []
109     dstliste.append(seg_d)
110     self[srckey] = dstliste
111
112 #}
113
114 class WoerterbuchVonDateien(Woerterbuch): #{
115
116     def __init__(self, quellsprache, zielsprache): #{
117         Woerterbuch.__init__(self, quellsprache, zielsprache)
118         self.lemma2src = Lemma2srcVonDateien(self, quellsprache, zielsprache)
119         self.src2dst = Src2dstVonDateien(self, quellsprache, zielsprache)
120         for wbdatei in config.wbdateien: #{
121             self.addWB(wbdatei)
122         #}
123     #}
124
125     def addWB(self, wbdatei): #{
126         self.p_begin("addWB")
127         if wbdatei.has_key(self.quellsprache) \
128         and wbdatei.has_key(self.zielsprache): #{
129             src = dokument.DokumentVonDatei(
130                 wbdatei[self.quellsprache]).getRegionenMit(segment.Segment)
131             dst = dokument.DokumentVonDatei(
132                 wbdatei[self.zielsprache]).getRegionenMit(segment.Segment)
133             anzahl = len(src)
134             if anzahl > len(dst): #{
135                 sys.stderr.write('Woerterbuch ' + 'wbdatei' + ' weniger Segmente ' + \
136                     'in der Sprache "' + self.zielsprache + '" als ' + \
137                     'in der Sprache "' + self.quellsprache + '".\n')
138             anzahl = len(dst)
139         #}
140         sys.stderr.write("Verarbeite " + 'anzahl' + " Segmente:\n")
141         letzteAusgabe = 0
142         for i in range(0,anzahl): #{
143             if time.time() - letzteAusgabe >= 1.0:
144                 sys.stderr.write('100*i/anzahl' + "% erledigt\r")
145                 letzteAusgabe = time.time()
146             seg_s = src[i]
147             seg_d = dst[i]
148             self.lemma2src.addSegmentpaar(seg_s, seg_d)
149             self.src2dst.addSegmentpaar(seg_s, seg_d)
150         #}
151         sys.stderr.write("100% erledigt\n")
152         self.p_end("addWB")
153         return 1
154     #}
155     self.p_end("addWB")
156     return 0
157 #}
158
159 def debug(self):
160     print "### lemma2src ###"
161     for i in self.lemma2src.keys():

```

```

162     print ("[" + 'i' + "]: " + `self.lemma2src[i]`)
163     print "### src2dst ###"
164     for i in self.src2dst.keys():
165         print ("[" + 'i' + "]: " + `self.src2dst[i]`)
166 #}
167
168 #####
169 #
170 # VonDb
171
172 class WoerterbuchVonDB(Woerterbuch): #{
173
174     def __init__(self, quellsprache, zielsprache): #{
175         Woerterbuch.__init__(self, quellsprache, zielsprache)
176         self.dbq = qclient.Query()
177         if not self.dbq.ok():
178             sys.stderr.write("Query-Server nicht gefunden!\n")
179
180     def close(self):
181         self.dbq.close()
182
183     def has_key(self, key):
184         self.p_begin("has_key")
185         self.dbq.setwbkorporus()
186         q = self.dbq.segment(match=key, lang=config.lang2id[self.quellsprache])
187         if q:
188             retval = len(q)
189         else:
190             retval = 0
191         self.p_end("has_key")
192         return retval
193
194     def __getitem__(self, woerter):
195         self.p_begin("getitem")
196         liste = []
197         self.dbq.setwbkorporus()
198         q = self.dbq.segment(match=woerter, lang=config.lang2id[self.quellsprache])
199         if q:
200             for seg in q:
201                 dst = seg.alignedRegion()
202                 liste.append((-1, seg, [dst]))
203         else:
204             f = open("wort_nicht_im_wb", "a")
205             f.write("%s\n" %woerter)
206             f.close()
207         self.p_end("getitem")
208         return liste
209
210 #}
211
212 class LemmatabuchVonDB(Woerterbuch): #{
213
214     def __init__(self, quellsprache, zielsprache, qdb_type="wb"): #{
215         Woerterbuch.__init__(self, quellsprache, zielsprache)
216         self.dbq = qclient.Query()
217         self.dbq.setkorporus(qdb_type)
218         if not self.dbq.ok():
219             sys.stderr.write("Query-Server nicht gefunden!\n")
220         self.cache = cache.DefaultCache(self)
221
222     def close(self):

```



```

223     self.dbq.close()
224
225 def has_key(self, key):
226     self.p_begin("has_key")
227     if "<unknown>" in key:                # 2do
228         self.p_end("has_key")
229         return 0
230     lstr = string.join(key)
231     ckey = (0, lstr)
232     if self.cache.has_key(ckey):
233         retval = self.cache[ckey]
234         self.p_end("has_key")
235         return retval
236     for i in ",;=.!?@":    # Satzzeichen, @card@ , etc.
237         if i in lstr:
238             self.cache[ckey] = 0
239             self.p_end("has_key")
240             return 0
241     ckey2 = (3, lstr)
242     if self.cache.has_key(ckey2):
243         q = self.cache[ckey2]
244     else:
245         self.p_begin("has_key_query")
246         q = self.dbq.segment(match_lf = lstr,
247                               strict   = "true",
248                               max      = "1",
249                               lang     = config.lang2id[self.quellsprache])
250         self.p_end("has_key_query")
251         self.cache[ckey2] = q
252     if q:
253         retval = len(q)
254     else:
255         retval = 0
256     self.cache[ckey] = retval
257     self.p_end("has_key")
258     # print retval
259     return retval
260
261 def __getitem__(self, lemmata):
262     self.p_begin("getitem")
263     liste = []
264     if "<unknown>" in lemmata:            # 2do
265         self.p_end("getitem")
266         return liste
267     lstr = string.join(lemmata)
268     ckey = (1, lstr)
269     if self.cache.has_key(ckey):
270         retval = self.cache[ckey]
271         self.p_end("getitem")
272         return retval
273     for i in ",;=.!?@":    # Satzzeichen, @card@ , etc.
274         if i in lstr:
275             self.cache[ckey] = liste
276             self.p_end("getitem")
277             return liste
278     ckey2 = (2, lstr)
279     if self.cache.has_key(ckey2):
280         q = self.cache[ckey2]
281     else:
282         self.p_begin("getitem_query")
283         q = self.dbq.segment(match_lf = string.join(lemmata),

```

```

284             strict = "true",
285             max     = "100",
286             lang    = config.lang2id[self.quellsprache])
287         self.p_end("getitem_query")
288         self.cache[ckey2] = q
289     if q:
290         for seg in q:
291             dst = seg.alignedRegion()
292             liste.append((-1, seg, [dst]))
293     else:
294         f = open("lemma_nicht_im_wb", "a")
295         f.write("%s\n" %lemmata)
296         f.close()
297     self.cache[ckey] = liste
298     self.p_end("getitem")
299     return liste
300
301
302 def test():
303     regiondb.setDefaultDbconn(db.Db())
304     wb_deen = WoerterbuchVonDB("de", "en")
305     wb_ende = WoerterbuchVonDB("en", "de")
306     for (wb, liste) in [(wb_deen, ['Apfel', 'Baum', 'Apfelbaum']),
307                        (wb_ende, ['apple', 'tree', 'apple tree'])]:
308         print 'wb'
309         for i in liste:
310             print 'i'
311             for (pos, src, dstlist) in wb[i]:
312                 print '\tde:', src.getText()
313                 for j in dstlist:
314                     print '\ten:', j.getText()
315
316 from Region      import dokument, segment
317 from DatabaseAPI import qclient, regiondb, db, config
318
319 if __name__ == '__main__':
320     test()
321

```

filterwb.py

Ein Wörterbuch, mit dem man nach Tags die Relevanz eines Wortes ermitteln kann.

```

1  #!/usr/bin/python
2
3  # Modul filterwb
4
5  import zeitmessung
6  import base64
7  import pickle
8  import string
9  import sys
10 import types
11 # weitere imports nach den Klassendefinitionen
12
13 class Relevant: #{
14
15     def __init__(self, sprache):
16         self.sprache = sprache
17
18     def relevant(self, region):
19         return 1

```

```

20
21 def p_getSlices(self, regionen, atome):
22     return range(len(atome))
23
24 def __call__(self, regionen):
25     atome = regionen.getRegionenMit(atom.Atom)
26     liste = self.p_getSlices(regionen, atome)
27     # print liste
28     relliste = []
29     for slice in liste:
30         if self.relevant(atome[slice]):
31             relliste.append(slice)
32             #print "Slice", 'slice', "OK"
33         else:
34             #print "Slice", 'slice', "verworfen"
35             pass
36     return (atome, relliste)
37
38 #}
39
40 #
41 # Python 2.1 workaround for slice objects
42 #
43 # "Slice objects are now comparable but not hashable; this prevents
44 # dict[:] from being accepted but meaningless."
45 # (Python 2.1 Release Notes)
46 #
47
48 import UserDict
49
50 class Dictionary(UserDict.UserDict): #{
51
52     def __init__(self):
53         UserDict.UserDict.__init__(self)
54         self.okey = {}
55
56     def __getitem__(self, key):
57         if type(key) == types.SliceType:
58             mkey = ('Slice', key.start, key.stop, key.step)
59         else:
60             mkey = key
61         return UserDict.UserDict.__getitem__(self, mkey)
62
63     def keys(self):
64         mkeys = UserDict.UserDict.keys(self)[: ] # copy list of keys
65         for index in range(len(mkeys)):
66             try:
67                 okey = self.okey[mkeys[index]]
68                 mkeys[index] = okey
69             except KeyError:
70                 pass
71         return mkeys
72
73     def __setitem__(self, key, value):
74         if type(key) == types.SliceType:
75             mkey = ('Slice', key.start, key.stop, key.step)
76         else:
77             mkey = key
78         self.okey[mkey] = value
79         UserDict.UserDict.__setitem__(self, mkey, value)
80

```

```

81     def has_key(self, key):
82         if type(key) == types.SliceType:
83             mkey = ('Slice', key.start, key.stop, key.step)
84         else:
85             mkey = key
86         return UserDict.UserDict.has_key(self, mkey)
87     #}
88
89
90 class WoerterbuchFilter(zeitmessung.Zeitmessung): #{
91
92     def __init__(self, quellsprache, zielsprache, relevant, qdb_type="wb"):
93         zeitmessung.Zeitmessung.__init__(self)
94         self.wb = wb.LemmatbuchVonDB(quellsprache, zielsprache, qdb_type)
95         self.quellsprache = quellsprache
96         self.zielsprache = zielsprache
97         self.relevant = relevant
98         self.cache = cache.DefaultCache(self)
99
100     def close(self):
101         self.wb.close()
102
103     def __repr__(self):
104         return "<WoerterbuchFilter %s zu %s>" %(self.quellsprache,
105                                             self.zielsprache)
106
107     def __getitem__(self, regionen):
108         self.p_begin("getitem")
109     #     print "getitem!", 'regionen'
110         if type(regionen) == types.TupleType: # neues Format?
111             format_neu = 1
112             markiert = regionen[0]           # Indizes der markierten Woerter
113             regionen = regionen[1]
114         else:
115             format_neu = 0
116         if self.cache.has_key(id(regionen)): # 2do: (w,t,l) benutzen?
117             retval = self.cache[id(regionen)]
118             self.p_end("getitem")
119             return retval
120         retval = Dictionary()
121         self.p_begin("relevant")
122         atome, slices = self.relevant(regionen)
123         if not format_neu: # altes Format?
124             markiert = range(len(atome)) # -> alles markiert
125         self.p_end("relevant")
126         self.p_begin("for bereich in slices")
127         vt = 'o'
128         for bereich in slices:
129             if type(bereich) == types.IntType:
130                 if vt == 'o':
131                     self.p_begin("type(bereich) == types.IntType")
132                     vt = 'i'
133                 elif vt == 's':
134                     self.p_end("type(bereich) != types.IntType")
135                     self.p_begin("type(bereich) == types.IntType")
136                     vt = 'i'
137                 index = bereich
138                 if not index in markiert:
139                     continue
140                 self.p_begin("atome[index]")
141                 einAtom = atome[index]

```

```

142 self.p_end("atome[index]")
143 if not retval.has_key(index):
144     self.p_begin("wb.has_key")
145     wbhaskey = self.wb.has_key(einAtom.getLemmata())
146     self.p_end("wb.has_key")
147     if wbhaskey:
148         self.p_begin("eintragsliste bestimmen")
149         eintragsliste = self.wb[einAtom.getLemmata()]
150         self.p_end("eintragsliste bestimmen")
151         segmente = {}
152         self.p_begin("eintragsliste durchgehen")
153         for (pos, srcseg, dstsegliste) in eintragsliste:
154             for dstseg in dstsegliste:
155                 if format_neu:
156                     segmente[srcseg.regionid] = (srcseg, dstseg)
157                 else:
158                     segmente[dstseg.getText()] = dstseg # unique machen
159         self.p_end("eintragsliste durchgehen")
160         if format_neu:
161             retval[index] = segmente.values()
162         else:
163             self.p_begin("Absatz bauen")
164             retval[index] = absatz.Absatz(segmente.values())
165             self.p_end("Absatz bauen")
166     else:
167         if format_neu:
168             retval[index] = []
169         else:
170             retval[index] = absatz.Absatz([]) # leerer Absatz
171 else:
172     if vt == 'o':
173         self.p_begin("type(bereich) != types.IntType")
174         vt = 's'
175     elif vt == 'i':
176         self.p_end("type(bereich) == types.IntType")
177         self.p_begin("type(bereich) != types.IntType")
178         vt = 's'
179     etwas_markiert = 0
180     for index in range(bereich.start, bereich.stop):
181         if index in markiert:
182             etwas_markiert = 1
183             break
184     if not etwas_markiert:
185         continue
186     atomliste = atome[bereich]
187     if not retval.has_key(bereich):
188         if self.wb.has_key(atomliste.getLemmata()):
189             eintragsliste = self.wb[atomliste.getLemmata()]
190             segmente = {}
191             self.p_begin("eintragsliste durchgehen")
192             for (pos, srcseg, dstsegliste) in eintragsliste:
193                 for dstseg in dstsegliste:
194                     if format_neu:
195                         segmente[srcseg.regionid] = (srcseg, dstseg)
196                     else:
197                         segmente[dstseg.getText()] = dstseg # unique machen
198             self.p_end("eintragsliste durchgehen")
199             if format_neu:
200                 retval[bereich] = segmente.values()
201             else:
202                 self.p_begin("Absatz bauen")

```

```

203         retval[bereich] = absatz.Absatz(segmente.values())
204         self.p_end("Absatz bauen")
205     else:
206         if format_neu:
207             retval[bereich] = []
208         else:
209             retval[bereich] = absatz.Absatz([]) # leerer Absatz
210     if vt == 's':
211         self.p_end("type(bereich) != types.IntType")
212     elif vt == 'i':
213         self.p_end("type(bereich) == types.IntType")
214     self.p_end("for bereich in slices")
215 #     print retval
216     self.cache[id(regionen)] = retval
217     self.p_end("getitem")
218     return retval
219     # Format: { 'atom1' : [seg11, seg12, ..., seg1n], ([] als Absatz)
220     #           'atom4' : ... } # nur relevante atomi
221     # wobei atomi = unique atoms in regionen
222     #           segij = WB-Eintrag zu lemma(atomi)
223
224     def has_key(self, regionen):
225         return 1
226 #}
227
228 class RelevantMax(Relevant): #{
229
230     def __init__(self, max, sprache):
231         Relevant.__init__(self, sprache)
232         self.max = max
233
234     def p_getSlices(self, regionen, atome):
235         liste = Relevant.p_getSlices(self, regionen, atome)
236         for startIndex in range(len(atome)):
237             ende = startIndex+self.max
238             if ende >= len(atome):
239                 ende = len(atome)-1
240             for endIndex in range(startIndex+2, ende+1):
241                 liste.append(slice(startIndex, endIndex))
242         return liste
243
244 #}
245
246 class RelevantNachTagP(RelevantMax): #{
247
248     def __init__(self, sprache, schwelle = 0, max = 4):
249         RelevantMax.__init__(self, max, sprache)
250         self.positiv = tags[config.lang2id[sprache]]
251         self.schwelle = schwelle
252
253     def relevant(self, region):
254         tags = region.getTags()
255         rel = 0
256         for tag in tags:
257             if self.positiv.has_key(tag):
258                 if self.positiv[tag] == -2: #alles, was mit -2 markiert ist, raus
259                     return 0.0
260                 rel = rel + self.positiv[tag]
261 #     print rel
262     #output = open("tags_all_"+self.sprache+".out", "a")
263     #output.write(string.join(tags, '\t') + '\n')

```

```

264         #output.close()
265         return rel > (self.schwelle * len(tags))
266
267     #}
268
269     class RelevantWegenNomen(RelevantMax): #{
270
271         def __init__(self, sprache, schwelle = 0, max = 4):
272             RelevantMax.__init__(self, max, sprache)
273             self.schwelle = schwelle
274
275         def relevant(self, region):
276             tags = region.getTags()
277             rel = 0
278             for tag in tags:
279                 if tag[0] != 'N': #alles, was nicht mit N anfaengt, raus
280                     return 0.0
281                 rel = rel + 2
282             return rel > (self.schwelle * len(tags))
283
284
285     class RelevantInseln(RelevantNachTagP): #{
286
287         def __init__(self, sprache, schwelle = 0):
288             self.inseltags = self.p_loadTags(sprache, config.tags)
289             RelevantNachTagP.__init__(self, sprache, schwelle, max(self.inseltags.values()))
290
291         def p_getSlices(self, regionen, atome):
292             liste = RelevantNachTagP.p_getSlices(self, regionen, atome)
293             indizes = liste[:]
294             for sl in indizes:
295                 if not self.inseltags.has_key(string.join(atome[sl].getTags())):
296                     liste.remove(sl)
297             return liste
298
299         def p_loadTags(self, sprache, dir):
300             tags = {}
301             tagfile = open(dir + "/tags_" + sprache + ".txt", "r")
302             for line in tagfile.readlines():
303                 tagline = string.strip(line)
304                 length = len(string.split(tagline))
305                 tags[tagline] = length
306             return tags
307
308     #}
309
310
311     class WoerterbuchFilterNachTagPOhneInseln(WoerterbuchFilter): #{
312
313         def __repr__(self):
314             return WoerterbuchFilter.__repr__(self)[-1] + \
315                 ", der nur Atome mit positivem Tag durchlaesst>"
316
317         def __init__(self, quellsprache, zielsprache, schwelle=0, qdb_type="wb"):
318             relevant = RelevantNachTagP(quellsprache, schwelle)
319             WoerterbuchFilter.__init__(self, quellsprache, zielsprache,
320                                         relevant, qdb_type)
321     #}
322
323     class WoerterbuchFilterNachTagP(WoerterbuchFilter): #{
324

```

```

325     def __repr__(self):
326         return WoerterbuchFilter.__repr__(self)[-1] + \
327             ", der nur Atome mit positivem Tag durchlaesst>"
328
329     def __init__(self, quellsprache, zielsprache, schwelle=0, qdb_type="wb"):
330         relevant = RelevantInseln(quellsprache, schwelle)
331         WoerterbuchFilter.__init__(self, quellsprache, zielsprache,
332             relevant, qdb_type)
333 #}
334
335
336 class RelevantNachTagN(RelevantMax): #{
337
338     def __init__(self, sprache, schwelle = 0, max = 4):
339         RelevantMax.__init__(self, max, sprache)
340         self.negativ = tags[config.lang2id[sprache]]
341         self.schwelle = schwelle
342
343     def relevant(self, atom):
344         tags = atom.getTags()[0]           # ??? atom ist beliebige region !!!
345         rel = 0
346         for tag in tags:
347             if self.negativ.has_key(tag):
348                 rel = rel + self.negativ[tag]
349         return rel > self.schwelle
350 #}
351
352 class WoerterbuchFilterNachTagN(WoerterbuchFilter):
353
354     def __repr__(self):
355         return WoerterbuchFilter.__repr__(self)[-1] + \
356             ", der nur Atome mit negativem Tag ausschliesst>"
357
358     def __init__(self, quellsprache, zielsprache, schwelle=0, qdb_type="wb"):
359         relevant = RelevantNachTagN(quellsprache, schwelle)
360         WoerterbuchFilter.__init__(self, quellsprache, zielsprache,
361             relevant, qdb_type)
362
363
364 def test():
365     regiondb.setDefaultDbconn(db.Db())
366     wbf_p = WoerterbuchFilterNachTagP("de", "en")
367     wbf_n = WoerterbuchFilterNachTagN("de", "en")
368     #reg1 = segmentdb.SegmentVonDb(160000, db.Db(), "1")
369     import tagger
370     reg1 = tagger.tagline("Wir müssen jemanden in Kenntnis setzen.", "de")
371     reg2 = tagger.tagline("Er kann nur mit Mühe und Not mit der Schreibmaschine schreiben.", "de")
372     for (wbf, reg) in [(wbf_p, reg1), (wbf_p, reg2),
373         (wbf_n, reg1), (wbf_n, reg2)]:
374         print "\n", 'wbf', "\n"
375         print reg.getText()
376         print reg.getTags()
377         d = wbf[reg]
378         a = reg.getRegionenMit(atom.Atom)
379         #print "a:", 'a'
380         #print "d:", 'd'
381         for key in d.keys():
382             #print 'Slice:', 'key'
383             print '\tde:', a[key].getText()
384             print '\ten:', d[key].getText()
385     regiondb.defaultDbconn.close()

```



```
386
387
388 from DatabaseAPI import config, regiondb, db, atomdb, segmentdb, cache
389 from Region import atom, region, absatz
390 import wb
391
392 tags = { config.lang2id["de"] : { # vorlaeufige Klassifizierung von Philip
393                                     '$(': -2,
394                                     '$,': -2,
395                                     '$.': -2,
396                                     'ADJA': +1,
397                                     'ADJD': +1,
398                                     'ADV': 0,
399                                     'APPO': 0,
400                                     'APPR': 0,
401                                     'APPRART': 0,
402                                     'APZR': 0,
403                                     'ART': -2,
404                                     'CARD': +1,
405                                     'FM': -2,
406                                     'ITJ': 0,
407                                     'KOKOM': 0,
408                                     'KON': 0,
409                                     'KOUJ': -1,
410                                     'KOUS': 0,
411                                     'NE': +2,
412                                     'NN': +2,
413                                     'PAV': -1,
414                                     'PDAT': -1,
415                                     'PDS': -1,
416                                     'PIAT': -1,
417                                     'PIDAT': -1,
418                                     'PIS': -1,
419                                     'PPER': -1,
420                                     'PPOSAT': -1,
421                                     'PPOSS': -1,
422                                     'PRELAT': -1,
423                                     'PRELS': -1,
424                                     'PRF': -1,
425                                     'PTKA': -1,
426                                     'PTKANT': -1,
427                                     'PTKNEG': 0,
428                                     'PTKVZ': +2,
429                                     'PTKZU': -1,
430                                     'PTKZV': 0,
431                                     'PWAT': -1,
432                                     'PWAV': -1,
433                                     'PWS': -1,
434                                     'SATZ-P': -2,
435                                     'TRUNC': -2,
436                                     'VAFIN': +2,
437                                     'VAIMP': +2,
438                                     'VAINF': +2,
439                                     'VAPP': +2,
440                                     'VMFIN': +2,
441                                     'VMINF': +2,
442                                     'VMPP': +2,
443                                     'VVFIN': +2,
444                                     'VVIMP': +2,
445                                     'VVINF': +2,
446                                     'VVIZU': +2,
```

```

447         'VVPP': +2,
448         'XY': -2,
449         #           LQL - Relevantliste von Arno
450         'adj': +1,
451         'adv': 0,
452         'art': -2,
453         'imp': -2,
454         'itj': -2,
455         'kon': -2,
456         'nn': +2,
457         'prep': -1,
458         'pron': -1,
459         'v': +2,
460         'va': +2,
461         'vi': +2,
462         'vt': +2,
463     },
464     config.lang2id["en"] : { # vorlaeufige Klassifizierung von Philip
465         "'": -2,
466         '#': -2,
467         '$': -2,
468         '"': -2,
469         '(': -2,
470         ')': -2,
471         ',': -2,
472         '.': -2,
473         ':': -2,
474         'CC': 0,
475         'CD': +1,
476         'DT': -2,
477         'EX': -2,
478         'FW': -2,
479         'IN': -1,
480         'JJ': +1,
481         'JJR': +1,
482         'JJS': +1,
483         'LS': -2,
484         'MD': +2,
485         'NN': +2,
486         'NNP': +2,
487         'NNPS': +2,
488         'NNS': +2,
489         'NP': +2,
490         'NPS': +2,
491         'PDT': -2,
492         'PP': -1,
493         'PP$': -1,
494         'PRP': -1,
495         'RB': -1,
496         'RBR': -1,
497         'RBS': -1,
498         'RP': -1,
499         'SATZ-P': -2,
500         'SYM': -2,
501         'TO': -2,
502         'UH': 0,
503         'VB': +2,
504         'VBD': +2,
505         'VBG': +2,
506         'VBN': +2,
507         'VBP': +2,

```

```

508         'VBZ': +2,
509         'WDT': -1,
510         'WP': -1,
511         'WP$': -1,
512         'WRB': -1,
513         #           LQL - Relevantliste von Arno
514         'adj': +1,
515         'adv': 0,
516         'art': -2,
517         'imp': -2,
518         'itj': -2,
519         'kon': -2,
520         'nn': +2,
521         'prep': -1,
522         'pron': -1,
523         'v': +2,
524         'va': +2,
525         'vi': +2,
526         'vt': +2,
527     },
528 }
529
530
531 if __name__ == '__main__':
532     test()
533

```

O.2.4 Datenbankserver und Hilfsklassen

querydb.py

Der Datenbankserver.

```

1  #!/usr/bin/python
2
3  from Region import region, absatz, atom, satz
4  import db, atomdb, satzdb, segmentdb, absatzdb, regiondb
5  import dokumentdb, cache, secret, config      # in current package
6  import filterwb, tagger                       # top-level module
7  import zeitmessung
8
9  import base64, getopt, pickle, popen2
10 import zlib
11 import os
12 import string, sys, thread, time, types
13 import SocketServer
14 from socket import *
15 import select
16 import MySQLdb
17
18 debug_level = 0      # Wert kann von Query.__init__ veraendert werden
19
20 todo = [
21     "nicht nur q_obj cachen",
22     "where und from Klauseln systematisch erzeugen",
23     "LIMIT bei max verwenden",
24     "cache bei update: pruefen",
25     "join benutzen, z.B. " + ""
26         select distinct map_s_w.satzid
27         from tokens left join map_s_w using ( tokenid )
28         where tokens.name = "werden";
29     "" + "deutlich schneller, Achtung: jedoch" + ""

```

```

30     select distinct t.tokenid
31     from tokens t left join grundformen g using (grundformid)
32     where g.name = 'Kenntnis';
33     """ + "deutlich langsamer",
34 ]
35
36 ERROR = "error!"
37
38 class Command:
39     pass
40
41 class CommandDb(Command):
42
43     def __init__(self, dbconndic):
44         self.dbconn = dbconndic
45
46 class QueryBase(CommandDb):          # nur Basis von Query+, nicht von Query
47
48     def __init__(self, dbconn, line, usekorpora, args):
49         CommandDb.__init__(self, dbconn)
50         self.usage = "bad command!"
51         self.korpora = usekorpora
52         args.append("max")
53         self.params = self.checkLine(line, args)
54         self.query_type = ""
55         self.public_queries = ["wort", "grundform", "tag"]
56         self.public_selects = ["text"]
57         self.public_wheres = ["name", "lang", "lemma", "wort", "parse"]
58         if not self.params:
59             raise ERROR, self.usage
60
61     def close(self):
62         pass
63
64     def __repr__(self):
65         params = []
66         for key in self.params.keys():
67             params.append((key, self.params[key]))
68         params.sort()          # feste Reihenfolge
69         paramrep = 'params'
70         if string.find(paramrep, "%") > -1:
71             paramrep = string.replace(paramrep, "%", "%%")
72         return ("<Query %s %s %s" %(paramrep, self.korpora,
73             'self.query_type')) + " %s>"
74
75     def checkLine(self, line, args):
76         listenkeys = [ "match", "match_l", "match_f", "match_lf",
77             "lemmata", "lem_inv",
78             "tags",
79         ]
80         params = {}
81         for item in string.split(line, ","):
82             split = string.split(item, "=")
83             if len(split) != 2 or not split[0] in args:
84                 raise ERROR, self.usage
85             key = split[0]
86             value = split[1]
87             # bei match muss es eine Liste sein,
88             # sind aber nicht immer spaces
89             if key in listenkeys:
90                 nvalue = []

```

```

91         for wort in string.split(value):
92             nvalue.append(string.replace(wort, "'", "'")) # SQL
93         value = nvalue
94     else:
95         value = string.replace(value, "'", "'") # SQL
96     params[key] = value
97 if params.has_key("strict"):
98     temp = string.lower(params["strict"])
99     if temp in ["true", "yes", "y", "1"]:
100         params["strict"] = 1
101     else:
102         del params["strict"]
103 if not params.has_key("strict"):
104     for key in listenkeys:
105         if params.has_key(key):
106             params[key].sort() # feste Reihenfolge fuer cache
107 return params
108
109 def query(self, restricted, select):
110     if restricted:
111         access_denied = 0
112         if not self.query_type in self.public_queries:
113             access_denied = 1
114         for i in self.params.keys():
115             if i not in self.public_wheres:
116                 access_denied = 1
117             if not select in self.public_selects:
118                 access_denied = 1
119         if access_denied:
120             raise ERROR, "access denied!"
121     list = self.real_query()
122     max = len(list)
123     if self.params.has_key("max"):
124         max = int(self.params["max"])
125     return list, max
126
127 def real_query(self):
128     return []
129
130 def querydb(self, query):
131     debug("Query = %s\n" % 'query', 9)
132     list = self.dbconn["query"].executelist(query)
133     if list:
134         return self.loadids(list)
135     return []
136
137 def p_intersect(self, list1, list2): # changes order of list1 and list2
138     list3 = []
139     list1.sort()
140     list2.sort()
141     debug('self'+".p_intersect"+'(list1, list2)+'\n', 25)
142     index1 = 0
143     index2 = 0
144     try:
145         while 1:
146             while list1[index1] < list2[index2]:
147                 index1 = index1 + 1
148             while list2[index2] < list1[index1]:
149                 index2 = index2 + 1
150             if list1[index1] == list2[index2]:
151                 list3.append(list1[index1])

```

```

152         index1 = index1 + 1
153         index2 = index2 + 1
154     except IndexError:
155         pass
156     debug(" -> "+`list3`+"\n", 25)
157     return list3
158
159     def loadids(self, idlist):
160         return map(lambda x: db.unknown2string(x[0]), idlist)
161
162 class QueryWort(QueryBase):
163
164     def __init__(self, dbconn, line, usekorpora, type = "wort"):
165         QueryBase.__init__(self, dbconn, line, usekorpora, ["name"])
166         self.dbquery = "select distinct t.tokenid from tokens t"
167         if type == "wort":
168             self.dbquery = self.dbquery + " where t.name ="
169             self.query_type = "wort"
170         elif type == "grundform":
171             self.dbquery = self.dbquery + ", grundformen g where t.grundformid = g.grundformid and g.name ="
172             self.query_type = "grundform"
173
174     def __repr__(self):
175         return QueryBase.__repr__(self) %self.dbquery
176
177     def real_query(self):
178         list = self.querydb(self.dbquery + " '%s' " %self.params["name"])
179         if self.params.has_key("max"):
180             list = list[:int(self.params["max"])]
181         atomlist = []
182         for id in list:
183             atomlist.append(atomdb.AtomVonDb("", self.dbconn["regionen"], id))
184         return atomlist
185
186
187 class WoerterbuchfilterNachBedarf:
188
189     def __init__(self, qs, zs, fs, qdb_type_list):
190         self.param = {}
191         for qdb_type in qdb_type_list:
192             self.param[(qs, qdb_type)] = (zs, fs)
193             self.param[(zs, qdb_type)] = (qs, fs)
194         self.cache = cache.DefaultCache(self)
195
196     def close(self):
197         for key in self.cache.keys():
198             self.cache[key].close()
199
200     def __getitem__(self, key):
201         p = self.param[key]
202         ckey = (key, p)
203         if self.cache.has_key(ckey):
204             return self.cache[ckey]
205         retval = filterwb.WoerterbuchFilterNachTagPOhneInseln(key[0], p[0], p[1], key[1])
206         self.cache[ckey] = retval
207         return retval
208
209     def has_key(self, key):
210         return self.param.has_key(key)
211
212     def keys(self):

```

```

213     return self.param.keys()
214
215
216 class QueryPhrase(QueryBase):
217
218     def __init__(self, dbconn, line, usekorpora, args):
219         QueryBase.__init__(self, dbconn, line, usekorpora, args)
220         if self.params.has_key("_fs"):
221             fs = float(self.params["_fs"])
222         else:
223             fs = 0.45
224         self.filter = WoerterbuchfilterNachBedarf("de", "en", fs, ["erkannt", "wb"])
225         self.stopwords = [
226             "not", "hat", "nicht", "as", "werden", "had", "with", "sich",
227             "dem", "das", "eine", "has", "by", "was", "mit", "will",
228             "is", "auf", "des", "im", "on", "an", "zu", "für",
229             "be", "von", "den", "that", "for", "a", "und", "and",
230             "die", "to", "der", "of", "in", "the",
231         ]
232         self.stoplemmata = [
233             "es", "from", "he", "this", "say", "bei", "nicht", "not",
234             "will", "at", "as", "with", "nach", "by", "mit", "auf",
235             "zu", "on", "an", "für", "im", "von", "haben", "that",
236             "werden", "for", "have", "sein", "und", "a", "and", "ein",
237             "to", "be", "of", "in", "the",
238         ]
239         self.langids = [-1] # von Querys mit alignten Regionen neu zu belegen
240
241     def close(self):
242         self.filter.close()
243
244     def __repr__(self):
245         return QueryBase.__repr__(self) % "Phrase"
246
247     def p_froms(self, liste):
248         dic = {
249             'g'      : "grundformen",
250             'm'      : "map_s_w",
251             'ph_c'   : "phrase_count",
252             'ph_e'   : "phrase_examples",
253             's'      : "saetze",
254             't'      : "tokens",
255             'tag_de' : "tagset_ims_de",
256             'tag_en' : "tagset_ims_en",
257         }
258         if not liste:
259             raise ValueError, "keine Tabelle fuer from angegeben"
260         erledigt = []
261         s = ""
262         for i in liste:
263             if not i in erledigt:
264                 s = s + dic[i] + " " + i + ", "
265                 erledigt.append(i)
266         return "from " + s[:-2] + " "
267
268     def p_distinct(self, idliste): # MySQL distinct seltsam bei group und having
269         idliste.sort()
270         isfirst = 1
271         last = 0
272         retval = []
273         for value in idliste:

```

```

274         if isfirst or value != last:
275             retval.append(value)
276         last = value
277         isfrist = 0
278     return retval
279
280 def p_query(self, column):
281     # Achtung: nicht jede column - params Kombination funktioniert !
282     froms = []
283     if column[:2] == "s." or self.korpora != "":
284         froms.append('s')
285     list = []
286     if self.params.has_key("id"):
287         list = [self.params["id"]]
288         # 2do: parse und andere params: behandeln oder usage
289     elif self.params.has_key("satzid"):
290         froms.append('m')
291         list = self.querydb(
292             ("select distinct %s " %column) +\
293             self.p_froms(froms) +\
294             ("where m.satzid = %s " %self.params["satzid"]) +\
295             ("%s" %self.korpora)
296         )
297         # 2do: parse und andere params: behandeln oder usage
298     elif self.params.has_key("wort"):
299         froms.append('m')
300         froms.append('t')
301         list = self.querydb(
302             ("select distinct %s " %column) +\
303             self.p_froms(froms) +\
304             ("where t.name = '%s' " %self.params["wort"]) +\
305             ("and t.tokenid = m.tokenid ") +\
306             ("%s" %self.korpora)
307         )
308     elif self.params.has_key("lemma"):
309         froms.append('m')
310         froms.append('t')
311         froms.append('g')
312         if self.params.has_key("parse"):
313             froms.append('s')
314             ew = "and m.satzid = s.satzid " %s
315             pval = self.params["parse"]
316             if pval == "NULL":
317                 ew = ew + "and s.parsequalitaet IS NULL "
318             elif pval[:2] == "GT":
319                 ew = ew + "and s.parsequalitaet > %s " %pval[2:]
320             else:
321                 ew = ew + "and s.parsequalitaet = %s " %pval
322         else:
323             ew = ""
324         list = self.querydb(
325             ("select distinct %s " %column) +\
326             self.p_froms(froms) +\
327             ("where g.name = '%s' " %self.params["lemma"]) +\
328             ("and t.tokenid = m.tokenid ") +\
329             ew +\
330             ("and g.grundformid = t.grundformid %s" %self.korpora)
331         )
332     elif self.params.has_key("match"):
333         phrasewords = self.params["match"][:]
334         for word in phrasewords[:]:

```



```

335         if word in self.stopwords:
336             phrasewords.remove(word)
337     if not phrasewords:
338         for stopword in self.stopwords:
339             if stopword in self.params["match"]:
340                 phrasewords = [stopword]
341                 break
342     if not phrasewords:
343         raise ERROR, "match ohne Wert!"
344     phraselen = len(phrasewords)
345     phrase= string.join(phrasewords)
346     phrasewds = "("+string.join(phrasewords, "','")+'"')
347     froms.append('m')
348     froms.append('t')
349     if self.params.has_key("parse"):
350         froms.append('s')
351         ew = "and s.satzid = m.satzid "
352         pval = self.params["parse"]
353         if pval == "NULL":
354             ew = ew + "and s.parsequalitaet IS NULL "
355         elif pval[:2] == "GT":
356             ew = ew + "and s.parsequalitaet > %s " %pval[2:]
357         else:
358             ew = ew + "and s.parsequalitaet = %s " %pval
359     else:
360         ew = ""
361     list = self.querydb(
362         ("select distinct %s " %column) +\
363         self.p_froms(froms) +\
364         ("where t.name IN %s " %phrasewds) +\
365         ("and t.tokenid = m.tokenid ") +\
366         ("%s " %self.korpora) +\
367         ew +\
368         ("group by m.satzid ") +\
369         ("having count(m.tokenid) >= %s" %phraselen)
370     )
371     elif self.params.has_key("match_f"):
372         phrasewords = self.params["match_f"][:] # list
373         wlen = len(phrasewords)
374         wkey = [1, column, self.korpora] # muss sich von lkey unterscheiden
375         if self.params.has_key("parse"):
376             wkey.append(self.params["parse"])
377         else:
378             wkey.append(None)
379         wkey = tuple(wkey)
380         for word in phrasewords[:]:
381             if word in self.stopwords \
382             and not self.dbconn["cache"].has_key((wkey, word)):
383                 phrasewords.remove(word)
384     if not phrasewords:
385         for stopword in self.stopwords:
386             if stopword in self.params["match_f"]:
387                 phrasewords = [stopword]
388                 break
389     if not phrasewords:
390         raise ERROR, "match_f ohne Wert!"
391     froms.append('m')
392     froms.append('t')
393     if self.params.has_key("parse"):
394         froms.append('s')
395         ew = "and m.satzid = s.satzid "

```

```

396         pval = self.params["parse"]
397         if pval == "NULL":
398             ew = ew + "and s.parsequalitaet IS NULL "
399         elif pval[:2] == "GT":
400             ew = ew + "and s.parsequalitaet > %s " %pval[2:]
401         else:
402             ew = ew + "and s.parsequalitaet = %s " %pval
403     else:
404         ew = ""
405     for word in phrasewords:
406         debug("word%s\n" %'(column, word, self.korpora)', 16)
407         if self.dbconn["cache"].has_key((wkey, word)):
408             currentids = self.dbconn["cache"][(wkey, word)]
409             debug("currentids aus cache\n", 16)
410         else:
411             currentids = self.querydb(
412                 ("select distinct %s " %column)+\
413                 self.p_frams(frams) +\
414                 ("where t.name = '%s' " %word) +\
415                 ("and t.tokenid = m.tokenid ") +\
416                 ew +\
417                 ("%s" %self.korpora)
418             )
419             debug("currentids aus db\n", 16)
420             self.dbconn["cache"][(wkey, word)] = currentids
421         # currentids reduzieren auf Regionen, die nicht zu viele
422         # Tokens enthalten
423         if string.strip(column) in ["m.segmentnr"]:
424             offset = 0
425             csl = currentids[offset:offset+100]
426             ncsl = []
427             while csl:
428                 ncsl = ncsl + self.p_distinct(self.querydb(
429                     ("select %s " %column) +\
430                     ("from map_s_w m ") +\
431                     ("where m.segmentnr in (%s) " %('csl'[1:-1])) +\
432                     ("group by m.satzid ") +\
433                     ("having count(m.tokenid) = %s" %wlen)
434                 ))
435                 offset = offset + 100
436                 csl = currentids[offset:offset+100]
437             currentids = ncsl
438             debug("currentids reduziert\n", 16)
439             debug("currentids = %s\n" %'currentids', 18)
440         else:
441             debug("currentids nicht reduziert\n", 16)
442         # currentids mit vorhandener Liste schneiden
443         if list:
444             list = self.p_intersect(currentids, list)
445             debug("geschnitten: %s\n" %'list', 18)
446         else:
447             list = currentids           # erster Durchlauf
448             if not list:                 # leerer Durchschnitt?
449                 debug("Schnitt bereits leer\n", 16)
450                 break                   # -> Ergebnis leer
451     elif self.params.has_key("match_1"):
452         phrasewords = self.params["match_1"][:]
453         for word in phrasewords[:]:
454             if word in self.stoplemmata:
455                 phrasewords.remove(word)
456     if not phrasewords:

```

```

457     for stopword in self.stoplemmata:
458         if stopword in self.params["match_l"]:
459             phrasewords = [stopword]
460             break
461     if not phrasewords:
462         raise ERROR, "match_l ohne Wert!"
463     phraselen = len(phrasewords)
464     phrase= string.join(phrasewords)
465     phrasewds = "("+string.join(phrasewords, "','")+'"')
466     froms.append('g')
467     froms.append('m')
468     froms.append('t')
469     froms.append('s')
470     if self.params.has_key("parse"):
471         pval = self.params["parse"]
472         if pval == "NULL":
473             ew = "and s.parsequalitaet IS NULL "
474         elif pval[:2] == "GT":
475             ew = "and s.parsequalitaet > %s " %pval[2:]
476         else:
477             ew = "and s.parsequalitaet = %s " %pval
478     else:
479         ew = ""
480     list = self.querydb(
481         ("select distinct %s " %column) +\
482         self.p_froms(froms) +\
483         ("where g.name IN %s " %phrasewds) +\
484         ("and t.tokenid = m.tokenid ") +\
485         ("%s " %self.korpora) +\
486         ("and s.satzid = m.satzid ") +\
487         ("and t.grundformid = g.grundformid ") +\
488         ew +\
489         ("group by m.satzid ") +\
490         ("having count(m.tokenid) >= %s" %phraselen)
491     )
492     elif self.params.has_key("match_lf"):
493         lemmatas = self.params["match_lf"][:] # list
494         lemlen = len(lemmatas)
495         lkey = [column, self.korpora]
496         if self.params.has_key("parse"):
497             lkey.append(self.params["parse"])
498         else:
499             lkey.append(None)
500         lkey = tuple(lkey)
501         for word in lemmatas[:]:
502             if word in self.stoplemmata \
503                 and not self.dbconn["cache"].has_key((lkey, word)):
504                 lemmatas.remove(word)
505         if not lemmatas:
506             for stopword in self.stoplemmata:
507                 if stopword in self.params["match_lf"]:
508                     lemmatas = [stopword]
509                     break
510         if not lemmatas:
511             raise ERROR, "match_lf ohne Wert!"
512         froms.append('m')
513         froms.append('t')
514         froms.append('g')
515         if self.params.has_key("parse"):
516             froms.append('s')
517         ew = "and m.satzid = s.satzid "

```

```

518         pval = self.params["parse"]
519         if pval == "NULL":
520             ew = ew + "and s.parsequalitaet IS NULL "
521         elif pval[:2] == "GT":
522             ew = ew + "and s.parsequalitaet > %s " %pval[:2]
523         else:
524             ew = ew + "and s.parsequalitaet = %s " %pval
525     else:
526         ew = ""
527     for lemma in lemmatas:
528         debug("lemma%s\n" %'(column, lemma, self.korpora)', 16)
529         if self.dbconn["cache"].has_key((lkey, lemma)):
530             currentids = self.dbconn["cache"][(lkey, lemma)]
531             debug("currentids aus cache\n", 16)
532         else:
533             currentids = self.querydb(
534                 ("select distinct %s " %column)+\
535                 self.p_frams(frams) +\
536                 ("where g.name = '%s' " %lemma) +\
537                 ("and t.tokenid = m.tokenid ") +\
538                 ("and g.grundformid = t.grundformid ") +\
539                 ew +\
540                 ("%s" %self.korpora)
541             )
542             debug("currentids aus db\n", 16)
543             self.dbconn["cache"][(lkey, lemma)] = currentids
544             # currentids reduzieren auf Regionen, die nicht zu viele
545             # Tokens enthalten
546             if string.strip(column) in ["m.segmentnr"]:
547                 offset = 0
548                 csl = currentids[offset:offset+100]
549                 ncsl = []
550                 while csl:
551                     ncsl = ncsl + self.p_distinct(self.querydb(
552                         ("select %s " %column) +\
553                         ("from map_s_w m ") +\
554                         ("where m.segmentnr in (%s) " %('csl'[1:-1])) +\
555                         ("group by m.satzid ") +\
556                         ("having count(m.tokenid) = %s" %lemlen)
557                     ))
558                     offset = offset + 100
559                     csl = currentids[offset:offset+100]
560                 currentids = ncsl
561                 debug("currentids reduziert\n", 16)
562                 debug("currentids = %s\n" %'currentids', 18)
563             else:
564                 debug("currentids nicht reduziert\n", 16)
565             # currentids mit vorhandener Liste schneiden
566             if list:
567                 list = self.p_intersect(currentids, list)
568             else:
569                 list = currentids           # erster Durchlauf
570             if not list:                   # leerer Durchschnitt?
571                 debug("Schnitt bereits leer\n", 16)
572                 break                       # -> Ergebnis leer
573         elif self.params.has_key("lemmata")\
574         or self.params.has_key("lem_inv"):
575             lkey = [column, self.korpora]
576             if self.params.has_key("parse"):
577                 lkey.append(self.params["parse"])
578         else:

```

```

579     lkey.append(None)
580 lkey = tuple(lkey)
581 if self.params.has_key("lemmata"):
582     lemmatas = self.params["lemmata"][:] # list
583     for word in lemmatas[:]:
584         if word in self.stoplemmata \
585             and not self.dbconn["cache"].has_key((lkey, word)):
586             lemmatas.remove(word)
587     if not lemmatas:
588         for stopword in self.stoplemmata:
589             if stopword in self.params["lemmata"]:
590                 lemmatas = [stopword]
591                 break
592     if not lemmatas:
593         raise ERROR, "lemmata ohne Wert!"
594 else:
595     lemmatas = []
596 if self.params.has_key("lem_inv"):
597     lem_invs = self.params["lem_inv"] # list
598 else:
599     lem_invs = []
600 froms.append('m')
601 froms.append('t')
602 froms.append('g')
603 if self.params.has_key("parse"):
604     froms.append('s')
605     ew = "and m.satzid = s.satzid "
606     pval = self.params["parse"]
607     if pval == "NULL":
608         ew = ew + "and s.parsequalitaet IS NULL "
609     elif pval[:2] == "GT":
610         ew = ew + "and s.parsequalitaet > %s " %pval[:2]
611     else:
612         ew = ew + "and s.parsequalitaet = %s " %pval
613 else:
614     ew = ""
615 list_leer = 0
616 for lemma in lemmatas:
617     debug("lemma%s\n" %'(column, lemma, self.korpora)', 16)
618     if self.dbconn["cache"].has_key((lkey, lemma)):
619         currentids = self.dbconn["cache"][(lkey, lemma)]
620         debug("currentids aus cache\n", 16)
621     else:
622         currentids = self.querydb(
623             ("select distinct %s " %column)+\
624             self.p_froms(froms) +\
625             ("where g.name = '%s' " %lemma) +\
626             ("and t.tokenid = m.tokenid ") +\
627             ("and g.grundformid = t.grundformid ") +\
628             ew +\
629             ("%s" %self.korpora)
630         )
631         debug("currentids aus db\n", 16)
632         self.dbconn["cache"][(lkey, lemma)] = currentids
633     if list:
634         list = self.p_intersect(currentids, list)
635     elif not list_leer:
636         list = currentids # erster Durchlauf
637     if not list: # leerer Durchschnitt?
638         list_leer = 1
639     debug("Schnitt bereits leer\n", 16)

```

```

640         break                                # -> Ergebnis leer
641     for lem_inv in lem_invs:
642         debug("lem_inv%s\n" % `(column, lem_inv, self.korpora)`, 16)
643         if self.dbconn["cache"].has_key((lkey, lem_inv)):
644             currentids = self.dbconn["cache"][(lkey, lem_inv)]
645             debug("currentids aus cache\n", 16)
646         else:
647             currentids = self.querydb(
648                 ("select distinct %s " %column)+\
649                 self.p_froms(froms) +\
650                 ("where g.reverse LIKE '%s%' " %lem_inv) +\
651                 ("and t.tokenid = m.tokenid ") +\
652                 ("and g.grundformid = t.grundformid ") +\
653                 ew +\
654                 ("%s" %self.korpora)
655             )
656             debug("currentids aus db\n", 16)
657             self.dbconn["cache"][(lkey, lem_inv)] = currentids
658         if list:
659             list = self.p_intersect(currentids, list)
660         elif not list_leer:
661             list = currentids                # erster Durchlauf
662             if not list:                    # leerer Durchschnitt?
663                 list_leer = 1
664                 debug("Schnitt bereits leer\n", 16)
665                 break                        # -> Ergebnis leer
666     elif self.params.has_key("parse"):      # parse alleine
667         froms.append('s')
668         froms.append('m')
669         pval = self.params["parse"]
670         if pval == "NULL":
671             ew = "and s.parsequalitaet IS NULL "
672         elif pval[:2] == "GT":
673             ew = "and s.parsequalitaet > %s " %pval[2:]
674         else:
675             ew = "and s.parsequalitaet = %s " %pval
676         if self.params.has_key("max"):
677             limit = "LIMIT %s " %self.params["max"]
678         else:
679             limit = ""
680         list = self.querydb(
681             ("select distinct %s " %column) +\
682             self.p_froms(froms) +\
683             ("where m.satzid = s.satzid ") +\
684             ew +\
685             ("%s" %self.korpora) +\
686             limit
687         )
688     elif self.params.has_key("beispiel zum segment mit id") \
689     or self.params.has_key("bid"):
690         try:
691             if self.params.has_key("bid"):
692                 bid = self.params["bid"]
693             else:
694                 bid = self.params["beispiel zum segment mit id"]
695                 bid = int(bid)
696         except ValueError:
697             raise ERROR, "bid keine ganze Zahl"
698         if self.params.has_key("max"):
699             limit = "LIMIT %s " %self.params["max"]
700         else:

```

```

701     limit = ""
702     froms.append('m')
703     froms.append('ph_e')
704     froms.append('ph_c')
705     if self.params.has_key("parse"):
706         froms.append('s')
707         ew = "and m.satzid = s.satzid "
708         pval = self.params["parse"]
709         if pval == "NULL":
710             ew = ew + "and s.parsequalitaet IS NULL "
711         elif pval[:2] == "GT":
712             ew = ew + "and s.parsequalitaet > %s " %pval[2:]
713         else:
714             ew = ew + "and s.parsequalitaet = %s " %pval
715     else:
716         ew = ""
717     list = self.querydb(
718         ("select distinct %s " %column) +\
719         self.p_froms(froms) +\
720         ("where m.segmentnr = ph_e.segmentnr ") +\
721         ("and ph_e.pc_id = ph_c.pc_id ") +\
722         ("and ph_c.segmentnr = %s " %bid) +\
723         ew +\
724         ("%s" %self.korpora) +\
725         limit
726     )
727     elif self.params.has_key("phrases"):
728         try:
729             sent = self.params["phrases"]
730             lang = self.params["lang"]
731         except KeyError:
732             raise ERROR, "Sprache wurde nicht angegeben"
733         wort = ""
734         for m in ["wort", "interessant", "i"]:
735             wstart = string.find(sent, "<%s>" %m)
736             if wstart < 0:
737                 continue
738             wende = string.find(sent, "</%s>" %m, wstart)
739             if wstart >= 0 and wende >= 0:
740                 if wort:
741                     raise ERROR, "mehrere Markierungen nicht unterstuetzt"
742                 wstart = wstart + 2 + len(m)
743                 wort = string.strip(sent[wstart:wende])
744                 debug("\twort: "+wort+"\n", 10)
745             elif wstart >= 0:
746                 raise ERROR, "</%s> nach <%s> nicht gefunden" %(m,m)
747         tagged = self.p_tagline(sent, lang)
748         markiert = range(len(tagged.getWoerter())) # 2do
749         debug("\ttagged: "+tagged.getWoerter()+"\n",10)
750         for qdb_type in ["erkannt", "wb"]:
751             debug("\tqdb_type: "+qdb_type+"\n",10)
752             #global debug_level
753             #dl = debug_level
754             try:
755                 #debug_level = -5
756                 filter = self.filter[(lang, qdb_type)]
757                 #debug_level = dl
758             except KeyError:
759                 raise ERROR, "Sprache %s unbekannt" %lang`
760             #debug_level = -5
761             phrases = filter[(markiert, tagged)]

```

```

762         #debug_level = dl
763         debug("\tphrases.keys():"+`phrases.keys()`+"\n", 10)
764         atoms = tagged.getRegionenMit(atom.Atom)
765         for part in phrases.keys():
766             src_dst_liste = phrases[part]
767             if len(src_dst_liste) > 0:
768                 ausgangspkt = atoms[part]
769                 try:
770                     if ausgangspkt.getDelimiter() == atom.Atom.delimiter:
771                         ausgangspkt = region.Regionen([ausgangspkt])
772                 except AttributeError:
773                     pass
774                 debug("\tphrase:"+`ausgangspkt.getText()`+"\n", 10)
775                 if (not wort) \
776                 or (string.find(ausgangspkt.getText(), wort) >= 0):
777                     for (src, dst) in src_dst_liste:
778                         src.qdb_type = qdb_type
779                         list.append(src)
780                         debug("\tlist.append(%s)\n" `%src`, 20)
781                 #     ausgangspkt.qdb_type = qdb_type
782                 #     list.append(ausgangspkt)
783     elif self.params.has_key("tags"):
784         try:
785             lang = self.params["lang"]
786             max = int(self.params["max"])
787         except KeyError:
788             raise ERROR, "Sprache oder Max wurde nicht angegeben"
789         taglang = "tag_" + lang
790         lkey = (column, self.korpora)
791         tags = self.params["tags"][:]
792         froms.append('m')
793         froms.append('t')
794         froms.append(taglang)
795         while len(list) < max:
796             firsttag = tags[0]
797             offset = 0
798             idlist = self.querydb(
799                 ("select %s " %column)+\
800                 self.p_froms(froms) +\
801                 ("where %s.name = '%s' " %(taglang, firsttag)) +\
802                 ("and %s.tagid = t.tagid " %taglang) +\
803                 ("and t.tokenid = m.tokenid ") +\
804                 ("%s" %self.korpora) +\
805                 ("LIMIT %s, %s" %(offset, 2 * max))
806             )
807             offset = offset + (2 * max)
808             idlist = self.p_uniquify(idlist)
809             debug("currentids aus db%s\n" %idlist, 16)
810             for moretag in tags[1:]:
811                 idlist = self.querydb(
812                     ("select %s " %column)+\
813                     self.p_froms(froms) +\
814                     ("where %s.name = '%s' " %(taglang, moretag)) +\
815                     ("and %s.tagid = t.tagid " %taglang) +\
816                     ("and t.tokenid = m.tokenid ") +\
817                     ("and %s in (%s)" %(column, `idlist`[1:-1])) +\
818                     ("%s" %self.korpora)
819                 )
820                 idlist = self.p_uniquify(idlist)
821                 debug("currentids aus db%s\n" %idlist, 16)
822             list = list + idlist

```



```

823     debug("list: " + `list` + "\n", 18)
824     return list
825
826 def p_uniquify(self, list):
827     nlist = []
828     for item in list:
829         if item in nlist:
830             continue
831         nlist.append(item)
832     return nlist
833
834 def p_tagline(self, line, lang):
835     return tagger.tagline(line, lang)
836
837 def p_dropwords(self, list):
838     matchkey = ""
839     for key in ["match", "match_l", "match_f", "match_lf", "tags"]:
840         if self.params.has_key(key):
841             matchkey = key
842             break
843     strict = self.params.has_key("strict")
844     if self.params.has_key("lemmata"):
845         list = self.p_dropwords_lemmata(list, strict)
846     elif self.params.has_key("lem_inv"):
847         list = self.p_dropwords_lem_inv(list, strict)
848     elif self.params.has_key("tags"):
849         list = self.p_dropwords_tag(list, strict)
850     elif matchkey:
851         list = self.p_dropwords_match(list, matchkey, strict)
852     return list
853
854 def p_dropwords_lemmata(self, list, strict):
855     return self.p_dropwords_match(list, "lemmata", strict)
856
857 def p_dropwords_lem_inv(self, list, strict):
858     return self.p_dropwords_match(list, "lem_inv", strict)
859
860 def p_dropwords_tag(self, list, strict):
861     return self.p_dropwords_match(list, "tags", strict)
862
863 def p_dropwords_match(self, list, matchkey, strict):
864     phrasewords = self.params[matchkey]
865     debug("phrasewords=%s\n" % `phrasewords`, 11)
866     debug("matchkey   =%s\n" % `matchkey`,   12)
867     debug("strict     =%s\n" % `strict`,     12)
868     #if len(phrasewords) <= 1:
869     #    return list                # nichts zu droppen
870     # sehr wohl! bei match* sind die Segmente zu verwerfen, die
871     # zusaetzliche Woerter enthalten
872     neulist = []
873     segmentliste = []
874     for index in range(len(list)):
875         neu = index % len(self.langids)
876         segmentliste.append(list[index])
877         if neu == (len(self.langids)-1):
878             uebernehmen = 0
879             for segment in segmentliste:
880                 if matchkey in ["match", "match_f"]:
881                     words = segment.getWoerter()
882                 elif matchkey in ["match_l", "match_lf", "lemmata", "lem_inv"]:
883                     words = segment.getLemmata()

```

```

884     elif matchkey in ["tags"]:
885         words = segment.getTags()
886     else:
887         raise ERROR, "%s nicht vollstaendig implementiert" %matchkey
888     debug("words      =%s\n" %'words', 20)
889     verworfen = 0
890     if strict:
891         #todo: fuer lem_inv implementieren!
892         if matchkey in [
893             "match", "match_f", "match_l", "match_lf",
894             "lemmata", "tags"
895         ]:
896             strictphrase = string.join(phrasewords, " ")
897             wholephrase = string.join(words, " ")
898             debug("strict: %s, whole: %s" %(strictphrase, wholephrase), 99)
899             if string.find(wholephrase, strictphrase) == -1:
900                 verworfen = 1
901                 debug("verworfen wg. strict!\n", 12)
902                 break
903         else:
904             raise ERROR, "strict fuer %s nicht vollstaendig implementiert" %matchkey
905     # wenn durch strict durchgegangen, koennen immernoch
906     # zuviele Woerter drin sein, also neues if
907     if matchkey in ["match", "match_f", "match_l", "match_lf"]:
908         for overfl in words:
909             if not overfl in phrasewords:
910                 verworfen = 1
911                 debug("verworfen wg. overfl=%s\n" %overfl, 12)
912                 break
913     elif matchkey in ["lemmata", "tags"]:
914         for notwendig in phrasewords:
915             if not notwendig in words:
916                 verworfen = 1
917                 debug("verworfen wg. notw.=%s\n" %notwendig, 12)
918                 break
919     elif matchkey in ["lem_inv"]:
920         for notwendig in phrasewords:
921             n_inv = string.lower(reverse(notwendig))
922             n_in_words = 0
923             for lem in words:
924                 if string.find(string.lower(lem),
925                     string.lower(n_inv)) + len(n_inv) == len(lem):
926                     n_in_words = 1
927                     break
928             if not n_in_words:
929                 verworfen = 1
930                 debug("verworfen wg. notw.=%s\n" %notwendig, 12)
931                 break
932     else:
933         raise NotImplementedError
934     laenge_ausreichend = len(words) >= len(phrasewords)
935     if not verworfen and laenge_ausreichend:
936         debug("Uebernahme\n", 12)
937         uebernehmen = 1
938     if uebernehmen:
939         for segment in segmentliste:
940             neulist.append(segment)
941     segmentliste = []
942 for i in neulist:
943     debug("neulist[%s]=%s\n" %('i',i.getText()), 25)
944 return neulist

```

```

945
946
947 class QuerySatz(QueryPhrase):
948
949     def __init__(self, dbconn, line, usekorpora):
950         QueryPhrase.__init__(self, dbconn, line, usekorpora, ["id", "wort",
951             "lemma", "match", "match_f", "lemmata", "lem_inv", "parse", "setparse",
952             "match_l", "match_lf", "_fs", "tags", "strict",
953             "beispiel zum segment mit id", "bid"])
954         self.query_type = "satz"
955
956     def real_query(self):
957         list = self.p_query(" m.satzid ")
958         if self.params.has_key("max"):
959             list = list[:int(self.params["max"])]
960         if self.params.has_key("setparse"):           # Parsequalitaet setzen
961             value = self.params["setparse"]
962             for id in list:
963                 self.dbconn["query"].update(
964                     ("update saetze ") +\
965                     ("set parsequalitaet = %s " %value) +\
966                     ("where satzid = %s" %id`))
967                 self.dbconn["cache"].invalidate()
968             return "Parsequalitaet gesetzt"
969         satzlist = []
970         for id in list:
971             satzlist.append(satzdb.SatzVonDb(id, self.dbconn["regionen"]))
972         return self.p_dropwords(satzlist)
973
974
975 class QuerySegment(QueryPhrase):
976
977     def __init__(self, dbconn, line, usekorpora):
978         QueryPhrase.__init__(self, dbconn, line, usekorpora, ["id", "wort",
979             "lemma", "match", "match_f", "lemmata", "lem_inv", "lang", "satzid",
980             "phrases", "parse", "match_l", "match_lf", "_fs", "tags", "strict",
981             "beispiel zum segment mit id", "bid"])
982         self.query_type = "segment"
983         self.langids = config.lang2id.values()
984         langmap = {}
985         langmap["deutsch"] = config.lang2id["de"]
986         langmap["englisch"] = config.lang2id["en"]
987         if self.params.has_key("lang"):
988             langstr = self.params["lang"]
989             if langmap.has_key(langstr):
990                 self.langids = [langmap[langstr]]
991             elif config.lang2id.has_key(langstr):
992                 self.langids = [config.lang2id[langstr]]
993             elif langstr in config.lang2id.values():
994                 self.langids = [langstr]
995             else:
996                 raise ERROR, "querydb.QuerySegment: Unbekannte Sprache "+`langstr`
997
998     def real_query(self):
999         segmentlist = []
1000         list = self.p_query(" m.segmentnr ")
1001         if self.params.has_key("max"):
1002             list = list[:int(self.params["max"])]
1003         if self.params.has_key("phrases"):
1004             return list
1005         for id in list:

```

```

1006         for lang in self.langids:
1007             segmentlist.append(segmentdb.SegmentVonDb(id, self.dbconn["regionen"], lang))
1008         return self.p_dropwords(segmentlist)
1009
1010 class Query(zeitmessung.Zeitmessung):
1011
1012     def __init__(self, regionenDbconn=None, dbg=-1):
1013         zeitmessung.Zeitmessung.__init__(self)
1014         if dbg >= 0:
1015             global debug_level
1016             debug_level = dbg
1017         self.dbconn = {}
1018         self.dbconnCloseOnExit = []
1019         self.dbconn["query"] = db.Db()
1020         self.dbconnCloseOnExit.append("query")
1021         if regionenDbconn:
1022             self.dbconn["regionen"] = regionenDbconn
1023         else:
1024             self.dbconn["regionen"] = db.Db()
1025             self.dbconnCloseOnExit.append("regionen")
1026         self.closed = 0
1027         self.setCache(cache.DefaultCache(self))
1028         self.setServer(None)
1029         self.setClientID(-1)
1030         self.selects = [ "text", "pyob", "all", "id", "count",
1031                         "debug",
1032                         "_r", "_r80", "_lr", "_lr80",
1033                         "_s", "_s80", "_ls", "_ls80"
1034                     ]
1035         self.types = ["wort", "grundform", "satz", "segment"]
1036         self.usekorpora = ""
1037         self.ETB = "<DONE>\n"
1038         self.ERROR = "<ERROR>\n"
1039         self.unpack={}
1040         self.unpack[None] = UnpackError(self.ERROR, self.ETB) # immer ERROR
1041         self.unpack["all"] = UnpackListRepr(self.ERROR, self.ETB)
1042         self.unpack["count"] = UnpackCount(self.ERROR, self.ETB)
1043         self.unpack["debug"] = UnpackListDebug(self.ERROR, self.ETB)
1044         self.unpack["id"] = UnpackListID(self.ERROR, self.ETB)
1045         self.unpack["pyob"] = UnpackPyob(self.ERROR, self.ETB)
1046         self.unpack["text"] = UnpackText(self.ERROR, self.ETB)
1047         self.unpack["_lr"] = UnpackListRepr(self.ERROR, self.ETB)
1048         self.unpack["_lr80"] = UnpackListRepr80(self.ERROR, self.ETB)
1049         self.unpack["_ls"] = UnpackListString(self.ERROR, self.ETB)
1050         self.unpack["_ls80"] = UnpackListString80(self.ERROR, self.ETB)
1051         self.unpack["_r"] = UnpackRepr(self.ERROR, self.ETB)
1052         self.unpack["_r80"] = UnpackRepr80(self.ERROR, self.ETB)
1053         self.unpack["_s"] = UnpackString(self.ERROR, self.ETB)
1054         self.unpack["_s80"] = UnpackString80(self.ERROR, self.ETB)
1055
1056     def setCache(self, cache):
1057         self.cache = cache
1058         self.dbconn["cache"] = cache
1059
1060     def setServer(self, server):
1061         self.server = server
1062
1063     def setClientID(self, cid):
1064         self.cid = cid
1065
1066     def __getstate__(self):

```

```

1067     return (self.usekorpora,)
1068
1069 def close(self):
1070     if self.closed:
1071         sys.stderr.write("Error: trying to close query already closed\n")
1072     for dbconnKey in self.dbconnCloseOnExit:
1073         self.dbconn[dbconnKey].close()
1074         del self.dbconn[dbconnKey]    # damit spaeter Verwendung auffaellt
1075     self.dbconnCloseOnExit = []
1076     self.closed = 1
1077
1078 def p_makeQuery(self, type, where, restricted = 0):
1079     q_obj = None
1080     if type == "wort" or type == "grundform":
1081         q_obj = QueryWort(self.dbconn, where, self.usekorpora, type)
1082     elif type == "satz" and not restricted:
1083         q_obj = QuerySatz(self.dbconn, where, self.usekorpora)
1084     elif type == "segment" and not restricted:
1085         q_obj = QuerySegment(self.dbconn, where, self.usekorpora)
1086     return q_obj
1087
1088 def p_checkStart(self, starts, vars):
1089     for i in range(0, len(vars)):
1090         var = vars[i]
1091         stw = starts[i]
1092         if var[0:2] != stw:
1093             raise ERROR, "bad command!"
1094         vars[i] = var[2:]
1095     return vars
1096
1097 def query(self, line, restricted=0):
1098     usage = "bad command!"
1099     q_line = string.split(line, ";")
1100     if len(q_line) == 1 and q_line[0][0:2] == "u::":
1101         corpora = string.lower(q_line[0][2:])
1102         if corpora == "wb":
1103             corpora = config.wbids
1104         elif corpora == "erkannt":
1105             corpora = "(-1)"
1106         elif corpora == "wb+erkannt":
1107             corpora = config.wbids[:-1] + ", -1)"
1108         elif corpora == "good":
1109             corpora = config.goodcorpora
1110         elif corpora in ["none", "all"]:
1111             corpora = None
1112         elif corpora[:1] == "(" :
1113             if corpora[-1:] != ")":
1114                 return ("schliessende Klammer fehlt", None)
1115         else:
1116             return ("Korpusart %s unbekannt" %corpora, None)
1117         if corpora:
1118             self.usekorpora = "and s.satzid = m.satzid " + \
1119                 "and s.herkunftsids in %s" %(corpora)
1120             return ("using corpora %s" %corpora, "_s")
1121         else:
1122             self.usekorpora = ""
1123             return ("using all corpora", "_s")
1124     else:
1125         if len(q_line) != 3:
1126             return (usage, None)
1127         else:

```

```

1128         try:
1129             type, select, where = self.p_checkStart(["t:", "s:", "w:"], q_line)
1130             if not select in self.selects or not type in self.types:
1131                 return (usage, None)
1132             q_obj = self.p_makeQuery(type, where)
1133             qkey = 'q_obj'
1134             debug("\tqkey = %s\n" %qkey, 11)
1135             try:
1136                 temp = self.cache[qkey]
1137                 list, max = temp
1138                 # 2do: immer nur bisher laengste Liste cachen
1139                 #     -> 'max' aus q_obj.__repr__() entfernen
1140                 #for i in list:
1141                 #    if isinstance(i, regiondb.RegionVonDb) \
1142                 #    or isinstance(i, atomdb.AtomVonDb):
1143                 #        i.setDbconn(self.dbconn["query"])           # 2do
1144             except KeyError:
1145                 list, max = q_obj.query(restricted, select)
1146                 self.cache[qkey] = (list, max)
1147             except TypeError:
1148                 debug("Error: %s in cache\n" %'temp')
1149                 list, max = q_obj.query(restricted, select)
1150                 self.cache[qkey] = (list, max)
1151             q_obj.close()
1152             return (list[:max], select)
1153         except ERROR, data:
1154             return (data, None)
1155         except MySQLdb.OperationalError, data:
1156             return ("Datenbankfehler %s" %data, None)
1157
1158     def serve(self, port):
1159         debug('Query.serve() called, instead use QueryRequestHandler!\n')
1160         sock = socket(AF_INET, SOCK_STREAM)
1161         sock.bind(("", port))
1162         sock.listen(1)
1163         debug('Service started on %s\n' % time.ctime(time.time()))
1164         stop = 0
1165         while not stop:
1166             conn, addr = sock.accept()
1167             stop = thread.start_new_thread(self.run, (conn, addr))
1168             sock.close()           # Was mach sock.shutdown() ?
1169             debug('Service stopped on %s\n' %time.ctime(time.time()))
1170
1171     def info(self, beschreibung):
1172         self.server.infoMutex.acquire()
1173         self.server.info[self.cid] = beschreibung
1174         self.server.infoMutex.release()
1175
1176     def infodone(self):
1177         self.server.infoMutex.acquire()
1178         self.server.info[self.cid] = self.server.info[self.cid] + " - done"
1179         self.server.infoMutex.release()
1180
1181
1182     def serverJob(self, name, *param):           # Jobs, die der Server erledigen muss
1183         self.p_begin("serverJob")
1184         self.info("beauftrage Server mit %s" %name)
1185         job = [self.cid, name]
1186         for i in param:
1187             job.append(i)
1188         self.server.jobsMutex.acquire()

```

```

1189 self.server.jobs.append(job)
1190 self.server.jobsMutex.release()
1191 fertig = 0
1192 while not fertig:
1193     time.sleep(1.0)
1194     self.server.jobsMutex.acquire()
1195     fertig = 1
1196     for i in self.server.jobs:
1197         if len(i) and i[0] == job[0]:
1198             fertig = 0
1199             break
1200     self.server.jobsMutex.release()
1201 self.infodone()
1202 self.p_end("serverJob")
1203 return "%s done" %name          # 2do: Antwort vom Server holen
1204
1205 def cmd_debug(self, data, conn, restricted):
1206     self.p_begin("cmd_debug")
1207     self.info("command " + `data`)
1208     global debug_level
1209     if string.strip(data[5:]):
1210         debug_level = int(data[5:])
1211         query = "debug level set to %s" %`debug_level`
1212     else:
1213         query = "debug level is %s" %`debug_level`
1214     self.unpack["_s"](conn.send, query, restricted)
1215     self.unpack["_s"](debug, query, restricted)
1216     self.infodone()
1217     self.p_end("cmd_debug")
1218     return 1
1219
1220 def cmd_cache(self, data, conn, restricted):
1221     self.p_begin("cmd_cache")
1222     self.info("command " + `data`)
1223     if data == "cache.clear":
1224         self.cache.invalidate()
1225         query = "cache cleared"
1226         self.unpack["_s"](conn.send, query, restricted)
1227         self.unpack["_s"](debug, query, restricted)
1228     elif data[:19] == "cache.autoclear mem":
1229         mem = string.strip(data[19:])
1230         try:
1231             self.server.ac_mem = int(mem)
1232         except:
1233             query = "number expected, found %s" %mem
1234             self.server.ac_mem2 = 0
1235             query = "mem set to %s KB" %mem
1236             self.unpack["_s"](conn.send, query, restricted)
1237             self.unpack["_s"](debug, query, restricted)
1238     elif data[:20] == "cache.autoclear time":
1239         zeit = string.strip(data[20:])
1240         try:
1241             sekunden = int(zeit)
1242         except:
1243             query = "number expected, found %s" %zeit
1244         if sekunden < 60:
1245             query = "number of seconds must be greater than or equal to 60"
1246         else:
1247             self.server.ac_trdist = sekunden
1248             self.server.ac_tcdist = sekunden / 2
1249             query = "time set to %s seconds" %sekunden

```

```

1250         self.unpack["_s"](conn.send, query, restricted)
1251         self.unpack["_s"](debug, query, restricted)
1252     elif data[:10] == "cache.save":
1253         fname = "qdb_"+string.strip(data[10:])+".cache"
1254         try:
1255             f=open(fname, "w")
1256             pickle.dump(self.cache, f)
1257             f.close()
1258             query = "cache saved as %s" %fname
1259         except IOError:
1260             query = "could not write to %s" %fname
1261         self.unpack["_s"](conn.send, query, restricted)
1262         self.unpack["_s"](debug, query, restricted)
1263     elif data[:10] == "cache.load":
1264         fname = "qdb_"+string.strip(data[10:])+".cache"
1265         ncache = None
1266         query = "cache not changed"
1267         try:
1268             f=open(fname, "r")
1269             ncache = pickle.load(f)
1270             f.close()
1271         except IOError:
1272             query = "unable to read from %s" %fname
1273     if ncache:
1274         if not self.server:
1275             self.setCache(ncache)
1276             query = "%s.cache loaded from %s" %(self,fname)
1277         else:
1278             query = self.serverJob("setCache", ncache)
1279     self.unpack["_s"](conn.send, query, restricted)
1280     self.unpack["_s"](debug, query, restricted)
1281     elif data[:9] == "cache.add":
1282         fname = "qdb_"+string.strip(data[9:])+".cache"
1283         ncache = None
1284         query = "cache not changed"
1285         try:
1286             f=open(fname, "r")
1287             ncache = pickle.load(f)
1288             f.close()
1289         except IOError:
1290             query = "unable to read from %s" %fname
1291     if ncache:
1292         num = 0
1293         for key in ncache.keys():
1294             self.cache[key] = ncache[key]
1295             num = num + 1
1296         query = "%s items added to %s.cache from %s" %(num,self,fname)
1297     self.unpack["_s"](conn.send, query, restricted)
1298     self.unpack["_s"](debug, query, restricted)
1299     elif data[:12] == "cache.delete":
1300         query = "no line deleted"
1301         try:
1302             line = string.strip(data[12:])
1303             q_line = string.split(line, ";")
1304             type, select, where = self.p_checkStart(["t:", "s:", "w:"], q_line)
1305             q_obj = self.p_makeQuery(type, where)
1306             line = `q_obj`
1307             if self.cache.has_key(line):
1308                 del self.cache[line]
1309                 query = "line %s deleted from cache" %(line)
1310         column = None

```



```

1311     if type == "satz":
1312         column = " m.satzid "
1313     if type == "segment":
1314         column = " m.segmentnr "
1315     ckey = []
1316     if q_obj.params.has_key("match_f"):
1317         ckey.append(1)
1318     if q_obj.params.has_key("match_f") \
1319     or q_obj.params.has_key("match_lf"):
1320         ckey.append(column)
1321         ckey.append(q_obj.korpora)
1322         if q_obj.params.has_key("parse"):
1323             ckey.append(q_obj.params["parse"])
1324         else:
1325             ckey.append(None)
1326         ckey = tuple(ckey)
1327     if q_obj.params.has_key("lemmata") \
1328     or q_obj.params.has_key("lem_inv"):
1329         query = query + "\nWarnung: derived lines not deleted"
1330     words = []
1331     if q_obj.params.has_key("match_lf"):
1332         words = q_obj.params["match_lf"]
1333     if q_obj.params.has_key("match_f"):
1334         words = q_obj.params["match_f"]
1335     for word in words:
1336         if self.cache.has_key((ckey, word)):
1337             del self.cache[(ckey, word)]
1338             query = query + "\nderived line %s deleted" % (ckey, word)
1339     except ERROR:
1340         pass
1341     self.unpack["_s"](conn.send, query, restricted)
1342     self.unpack["_s"](debug, query, restricted)
1343     elif data == "cache.stat":
1344         query = self.cache.statistic()
1345         self.unpack["_s"](conn.send, query, restricted)
1346         self.unpack["_s"](debug, query, restricted)
1347     elif data == "cache.keys":
1348         query = self.cache.keys()
1349         self.unpack["_lr"](conn.send, query, restricted)
1350         self.unpack["_lr"](debug, query, restricted)
1351     elif data == "cache.items":
1352         query = []
1353         for (key, value) in self.cache.items():
1354             if type(key) == types.StringType:
1355                 kstr = key
1356             else:
1357                 kstr = 'key'
1358             if type(value) == types.ListType:
1359                 vstr = "IDs "+value'
1360             elif type(value) == types.TupleType:
1361                 try:
1362                     l,m = value
1363                     l2 = []
1364                     for reg in l:
1365                         l2.append(reg.regionid)
1366                     vstr = "Regs " + 'l2'
1367                 except:
1368                     vstr = 'value'
1369             else:
1370                 vstr = 'value'
1371         query.append(kstr+'\t'+vstr)

```

```

1372         self.unpack["_ls80"](conn.send, query, restricted)
1373         self.unpack["_ls"](debug, query, restricted)
1374     else:
1375         self.info("bad command " + `data`)
1376         self.infodone()
1377         self.p_end("cmd_cache")
1378         return 0
1379     self.infodone()
1380     self.p_end("cmd_cache")
1381     return 1
1382
1383 def cmd_info(self, data, conn, restricted):
1384     self.p_begin("cmd_info")
1385     self.info("command " + `data`)
1386     arg = string.strip(data[4:])
1387     if arg == "":
1388         self.server.infoMutex.acquire()
1389         query = self.server.info.keys()
1390         self.server.infoMutex.release()
1391         self.unpack["_r"](conn.send, query, restricted)
1392     else:
1393         arglst = string.split(arg)
1394         query = "Argument erwartet"
1395         if "*" in arglst:
1396             arglst = []
1397             self.server.infoMutex.acquire()
1398             for i in self.server.info.keys():
1399                 arglst.append(`i`)
1400             self.server.infoMutex.release()
1401             query = "kein Client-Thread vorhanden"
1402         if len(arglst) == 0:
1403             self.unpack[None](conn.send, query, restricted)
1404         elif len(arglst) == 1:
1405             self.server.infoMutex.acquire()
1406             try:
1407                 query = self.server.info[int(arglst[0])]
1408             except:
1409                 query = "no info for %s" %arglst[0]
1410             self.server.infoMutex.release()
1411             self.unpack["_s"](conn.send, query, restricted)
1412         else:
1413             query = []
1414             for arg in arglst:
1415                 self.server.infoMutex.acquire()
1416                 try:
1417                     query.append(arg + ": " + self.server.info[int(arg)])
1418                 except:
1419                     query.append(arg + ": " + "no info")
1420                 self.server.infoMutex.release()
1421             self.unpack["_ls"](conn.send, query, restricted)
1422     self.infodone()
1423     self.p_end("cmd_info")
1424     return 1
1425
1426 def cmd_status(self, data, conn, restricted):
1427     self.p_begin("cmd_status")
1428     self.info("command " + `data`)
1429     arg = string.strip(data[6:])
1430     slist = ["time", "dbname", "dbuser", "cwd", "user", "user_env",
1431             "dbconnCloseOnExit",
1432             "dbconn", "closed", "selects", "types", "usekorpora",

```

```

1433     "cache", "server", "cid",
1434     "server.start", "server.port", "version",
1435     "server.ac_tclast", "server.ac_trlast",
1436     "server.ac_tcdist", "server.ac_trdist",
1437     "server.ac_mem", "server.ac_mem2",
1438     "server.tlast", "server.tcount",
1439     "size", "pid",
1440 ]
1441 slist.sort()
1442 if arg == "":
1443     query = "OK"
1444     self.unpack["_s"](conn.send, query, restricted)
1445 elif not restricted:
1446     if arg == "*":
1447         args = slist
1448     elif arg in slist:
1449         args = [arg]
1450     elif arg == "db":
1451         args = ["dbname", "dbuser"]
1452     else:
1453         args = string.split(arg)
1454     query = []
1455     for arg in args:
1456         if arg == "time":
1457             query.append("time server:")
1458             self.server.zeitMutex.acquire()
1459             s = self.server.statistic()
1460             for key in s.keys():
1461                 query.append("%s: %s" %(key, `s[key]`))
1462             self.server.zeitMutex.release()
1463             query.append("time self:")
1464             s = self.statistic()
1465             for key in s.keys():
1466                 query.append("%s: %s" %(key, `s[key]`))
1467         elif arg == "version":
1468             global version
1469             query.append("version = %s" %version)
1470         elif arg == "cwd":
1471             query.append("cwd = '%s'" %os.getcwd())
1472         elif arg == "dbname":
1473             query.append("dbname = '%s'" %config.dbname)
1474         elif arg == "dbuser":
1475             query.append("dbuser = '%s'" %config.dbuser)
1476         elif arg == "user_env":
1477             query.append("user_env = '%s'" %os.environ["USER"])
1478         elif arg == "user":
1479             query.append("user = '%s'" %config.user)
1480         elif arg == "cache":
1481             ci = {}
1482             ci["num_keys"] = len(self.cache.keys())
1483             ci["id"] = id(self.cache)
1484             ci["statistic"] = self.cache.statistic()
1485             query.append("cache = '%s'" %`ci`)
1486         elif arg == "dbconn":
1487             ci = {}
1488             for key in self.dbconn.keys():
1489                 if key != "cache":
1490                     ci[key] = self.dbconn[key]
1491             query.append("dbconn = '%s'" %`ci`)
1492         elif arg == "size":
1493             size = getVmSize()

```

```

1494         query.append("size = %s" %size)
1495     elif arg == "pid":
1496         pid = os.getpid()
1497         query.append("pid = %s" %pid)
1498     elif arg in slist:
1499         obj = self
1500         for attr in string.split(arg, '.'):
1501             try:
1502                 obj = getattr(obj, attr)
1503             except AttributeError:
1504                 query.append("%s.%s not found" %('obj',attr))
1505         query.append("%s = %s" %(arg, 'obj'))
1506     else:
1507         debug("cmd_status %s unbekannt\n" %'arg', 10)
1508         self.infodone()
1509         self.p_end("cmd_status")
1510         return 0
1511     self.unpack["_ls"](conn.send, query, restricted)
1512 else:
1513     debug("cmd_status %s verweigert\n" %'arg', 10)
1514     self.infodone()
1515     self.p_end("cmd_status")
1516     return 0
1517 self.infodone()
1518 self.p_end("cmd_status")
1519 return 1
1520
1521 def run(self, conn, addr):
1522     self.p_begin("run")
1523     debug('Connected by %s\n' %'addr', -1)
1524     key = secret.getKey()
1525     conn.send("Was ist secret.getValue(" + 'key' + ")?\n")
1526     data = string.strip(conn.recv(1024))
1527     if not secret.valueOK(key, data):
1528         conn.send("Das ist leider falsch. "
1529                 "Sie erhalten nur Zugriff auf das Woerterbuch.\n")
1530         debug('\tAuthentication failed, restricted access only\n', 0)
1531         restricted = 1
1532     else:
1533         conn.send("OK\n")
1534         debug('\tAuthentication OK for %s\n' %'addr', 0)
1535         restricted = 0
1536     stop = 0
1537     while 1:
1538         self.p_begin("recv")
1539         self.info("waiting for command")
1540         data = conn.recv(1024)
1541         self.infodone()
1542         self.p_end("recv")
1543         self.p_begin("data")
1544         if not data :           # Client hat Verbindung geschlossen
1545             debug('Connection %s closed by client.\n' %'addr', -1)
1546             break
1547         data = string.strip(data)
1548         if not data:           # Zeilen nur mit Whitespace ignorieren
1549             continue
1550         debug(("\\t%s->" %'addr') + data + "\n", 3)
1551         if data == "quit":
1552             conn.send("bye\n%s" %self.ETB)
1553             break
1554         elif data == "shutdown" and not restricted:

```

```

1555     conn.send("stopping service\n")
1556     conn.send("this may take a few seconds\n%s" %self.ETB)
1557     stop = 1
1558     break
1559     elif data[:5] == "cache" and not restricted:
1560         if not self.cmd_cache(data, conn, restricted):
1561             self.p_begin("unpack")
1562             self.unpack[None](conn.send, "bad command", restricted)
1563             self.p_end("unpack")
1564     elif data[:5] == "debug" and not restricted:
1565         if not self.cmd_debug(data, conn, restricted):
1566             self.p_begin("unpack")
1567             self.unpack[None](conn.send, "bad command", restricted)
1568             self.p_end("unpack")
1569     elif data[:4] == "info" and not restricted:
1570         if not self.cmd_info(data, conn, restricted):
1571             self.p_begin("unpack")
1572             self.unpack[None](conn.send, "bad command", restricted)
1573             self.p_end("unpack")
1574     elif data[:6] == "status":
1575         if not self.cmd_status(data, conn, restricted):
1576             self.p_begin("unpack")
1577             self.unpack[None](conn.send, "bad command", restricted)
1578             self.p_end("unpack")
1579     else:
1580         self.info("query " + data)
1581         self.p_begin("query")
1582         query, select = self.query(data, restricted)
1583         self.p_end("query")
1584         self.info(data + " - unpacking")
1585         self.p_begin("unpack")
1586         self.unpack[select](conn.send, query, restricted)
1587         self.p_end("unpack")
1588         self.infodone()
1589         self.p_end("data")
1590     self.p_end("data")
1591     conn.close()
1592     debug('Connection to %s closed\n' %'addr', -1)
1593     self.info("closed")
1594     self.p_end("run")
1595     self.server.zeitMutex.acquire()
1596     self.server.p_addToStatistic(self.statistic())
1597     self.server.zeitMutex.release()
1598     return stop
1599
1600
1601 class Unpack:
1602
1603     def __init__(self, error_str, etb_str):
1604         self.ERROR = error_str
1605         self.ETB = etb_str
1606         self.SelectError = "SelectError"
1607
1608     def __call__(self, fun, query, restricted = 0):
1609         try:
1610             self.p_call(fun, query, restricted)
1611         except self.SelectError, data:
1612             apply(fun, ("%s" %self.ERROR,))
1613             apply(fun, ("%s\n" %data,))
1614         except:
1615             apply(fun, ("%s" %self.ERROR,))

```

```

1616         import traceback
1617         import StringIO
1618         tb = StringIO.StringIO()
1619         traceback.print_exc(file=tb)
1620         tb.seek(0)
1621         sys.stderr.write(tb.read())
1622         if not restricted:
1623             tb.seek(0)
1624             apply(fun, ("%s\n" %tb.read(),))
1625         else:
1626             apply(fun, ("exception caught\n",))
1627         tb.close()
1628         apply(fun, ("%s" %self.ETB,))
1629
1630     def p_call(self, fun, query, restricted):
1631         raise self.SelectError, "undefined select"
1632
1633
1634 class UnpackError(Unpack):
1635
1636     def p_call(self, fun, query, restricted):
1637         if type(query) == types.StringType:
1638             raise self.SelectError, query
1639         else:
1640             raise self.SelectError, `query`
1641
1642
1643 class UnpackPyob(Unpack):
1644
1645     def p_call(self, fun, query, restricted):
1646         objstr = pickle.dumps(query)
1647         obj64 = base64.encodestring(zlib.compress(objstr))
1648         if len(obj64) < len(objstr):
1649             compressed = 1
1650         else:
1651             compressed = 0
1652         obj64 = objstr # Achtung: Problem wenn ERROR oder ETB enthalten
1653         debug("\t\t-> pyob: sending "+`len(obj64)`+" chars\n", 10)
1654         apply(fun, ("%s\n" %compressed,))
1655         apply(fun, (obj64,))
1656
1657
1658 class UnpackCount(Unpack):
1659
1660     def p_call(self, fun, query, restricted):
1661         try:
1662             apply(fun, (`len(query)`+'`n',))
1663         except TypeError:
1664             raise self.SelectError, "Das Ergebnis %s hat keine Länge." %`query`
1665
1666
1667 class UnpackText(Unpack):
1668
1669     def p_call(self, fun, query, restricted, pr=1):
1670         if isinstance(query, region.Region):
1671             try:
1672                 #str = "<id:%s type:%s>\n" %(query.regionid, query.qdb_type)
1673                 apply(fun, ("<id:%s type:%s>\n" %(query.regionid, query.qdb_type), ))
1674             except AttributeError:
1675                 try:
1676                     apply(fun, ("<id:%s>\n" %query.regionid,))

```

```

1677         except AttributeError:
1678             pass
1679         #query.writeTagzeilen(fun)
1680         mm = map(None, query.getWoerter(),
1681                 query.getTags(),
1682                 query.getLemmata())
1683         for (w,t,l) in mm:
1684             apply(fun, ("%s\t%s\t%s\n" %(w,t,l),))
1685         try:
1686             delimiter = query.getDelimiter()
1687         except AttributeError:
1688             delimiter = "<unbestimmter Regionentyp>\n"
1689         apply(fun, ("%s" %delimiter,))
1690     elif type(query) == types.StringType:
1691         if not pr: # nur auf oberster Ebene erlaubt
1692             raise ValueError
1693         apply(fun, ("%s\n" %query,))
1694     elif type(query) == types.ListType:
1695         for i in query:
1696             self.p_call(fun, i, restricted, 0)
1697     else:
1698         if type(query) == types.InstanceType:
1699             what = query.__class__.__name__
1700         else:
1701             what = `type(query)`
1702         if what == "RegionenSlice": # warum notwendig?
1703             try:
1704                 apply(fun, ("<id:%s>\n" %query.regionid,))
1705             except AttributeError:
1706                 pass
1707             query.writeTagzeilen(fun)
1708         else:
1709             debug("\ttrying to output %s\n" %what)
1710             raise TypeError, "trying to output %s\n" %what
1711         debug("\t\t-> sent\n", 10)
1712
1713
1714     class UnpackItem(Unpack): # kombinieren mit IUHelp
1715
1716         def p_call(self, fun, query, restricted):
1717             self.p_item(fun, query, restricted)
1718
1719     class UnpackList(Unpack): # kombinieren mit IUHelp
1720
1721         def p_call(self, fun, query, restricted):
1722             #if isinstance(query, region.Regionen):
1723             #    apply(fun, ("<%s items in %s>\n" %('len(query)', 'query'),))
1724             if type(query) == types.ListType:
1725                 apply(fun, ("<%s items in list>\n" %'len(query)',))
1726             else:
1727                 self.p_item(fun, query, restricted)
1728                 return
1729             for i in query:
1730                 self.p_call(fun, i, restricted)
1731
1732     class IUHelp:
1733
1734         def p_item(fun, query, restricted):
1735             raise NotImplementedError
1736
1737     class IUHRepr(IUHelp):

```

```

1738
1739     def p_item(self, fun, query, restricted):
1740         apply(fun, ("%s\n" % 'query',))
1741
1742 class UnpackRepr(UnpackItem, IUHRepr): pass
1743 class UnpackListRepr(UnpackList, IUHRepr): pass
1744
1745
1746 class IUHRepr80(IUHelp):
1747
1748     def p_item(self, fun, query, restricted):
1749         repr = 'query'
1750         line = 'len(repr)' + ": " + repr
1751         apply(fun, ("%s\n" % line[:80],))
1752
1753 class UnpackRepr80(UnpackItem, IUHRepr80): pass
1754 class UnpackListRepr80(UnpackList, IUHRepr80): pass
1755
1756
1757 class IUHDebug(IUHelp):
1758
1759     def p_item(self, fun, query, restricted):
1760         apply(fun, ("type = %s\n" % 'type(query)',))
1761         if isinstance(query, region.Region):
1762             apply(fun, ("is instance of region.Region\n",))
1763             apply(fun, (".getWoerter = %s\n" % 'query.getWoerter()',))
1764         if isinstance(query, region.Regionen):
1765             apply(fun, ("is instance of region.Regionen\n",))
1766             apply(fun, (".subregionen = %s\n" % 'query.subregionen',))
1767         import segmentdb
1768         if isinstance(query, segmentdb.SegmentVonDb):
1769             apply(fun, ("is instance of segmentdb.SegmentVonDb\n",))
1770             apply(fun, (".getWoerter = %s\n" % 'query.getWoerter()',))
1771             apply(fun, (".subregionen = %s\n" % 'query.subregionen',))
1772             apply(fun, (".regionid = %s\n" % 'query.regionid',))
1773         apply(fun, ("dir = %s\n" % 'dir(query)',))
1774         for attr in dir(query):
1775             apply(fun, (".%s = %s\n" % (attr, 'getattr(query, attr)'),))
1776
1777
1778 class UnpackDebug(UnpackItem, IUHDebug): pass
1779 class UnpackListDebug(UnpackList, IUHDebug): pass
1780
1781
1782 class IUHString(IUHelp):
1783
1784     def p_item(self, fun, query, restricted):
1785         apply(fun, ("%s\n" % str(query),))
1786
1787 class UnpackString(UnpackItem, IUHString): pass
1788 class UnpackListString(UnpackList, IUHString): pass
1789
1790
1791 class IUHString80(IUHelp):
1792
1793     def p_item(self, fun, query, restricted):
1794         s = str(query)
1795         line = 'len(s)' + ": " + s
1796         apply(fun, ("%s\n" % line[:80],))
1797
1798 class UnpackString80(UnpackItem, IUHString80): pass

```



```

1799 class UnpackListString80(UnpackList, IUHString80): pass
1800
1801
1802 class IUHID(IUHHelp):
1803
1804     def p_item(self, fun, query, restricted):
1805         apply(fun, ("%s\n" % 'query.regionid',))
1806
1807 class UnpackID(UnpackItem, IUHID): pass
1808 class UnpackListID(UnpackList, IUHID): pass
1809
1810
1811 class QueryRequestHandler(SocketServer.BaseRequestHandler):
1812
1813     def __init__(self, request, client_address, server, qrhid, dbconn):
1814         self.id = qrhid
1815         self.dbconn = dbconn
1816         SocketServer.BaseRequestHandler.__init__(self, request,
1817                                                 client_address, server)
1818
1819     def setup(self):
1820         self.query = Query(self.dbconn)
1821         self.query.setCache(self.server.cache)
1822         self.query.setServer(self.server)
1823         self.query.setClientID(self.id)
1824
1825     def handle(self):
1826         try:
1827             if self.query.run(self.request, self.client_address):
1828                 # query will be closed in finish()
1829                 self.server.stop = 1 # stop server
1830         except:
1831             import traceback
1832             import StringIO
1833             tb = StringIO.StringIO()
1834             traceback.print_exc(file=tb)
1835             tb.seek(0)
1836             sys.stderr.write("\twhile handling %s:\n" % 'self.client_address')
1837             zeile = tb.readline()
1838             while zeile != "":
1839                 sys.stderr.write("\t"+zeile)
1840                 zeile = tb.readline()
1841
1842     def finish(self):
1843         self.query.close()
1844         self.server.qrhFinishedMutex.acquire()
1845         self.server.qrhFinished[self.id] = 1 # alter server-data
1846         self.server.qrhFinishedMutex.release()
1847
1848
1849 class QueryServer(SocketServer.ThreadingTCPServer, zeitmessung.Zeitmessung):
1850
1851     def __init__(self, port):
1852         zeitmessung.Zeitmessung.__init__(self)
1853         self.zeitMutex = thread.allocate_lock()
1854         self.cache = cache.MutexDefaultCache(self)
1855         #self.cache = cache.NullCache(self)
1856         self.qrhFinished = {}
1857         self.qrhFinishedMutex = thread.allocate_lock()
1858         self.nextqrhid = 0
1859         self.info = {}

```

```

1860     self.infoMutex = thread.allocate_lock()
1861     self.jobs = []
1862     self.jobsMutex = thread.allocate_lock()
1863     self.dbconn = db.Db()
1864     self.tcount = 0
1865     self.tlast = 0
1866     self.ac_tclast = 0          # autoclear-check: letzter tcount
1867     self.ac_tcdist = 60*5      # tcounts bis zum naechsten Check
1868     self.ac_trdist = 60*10     # Zeit bis zum naechsten Check
1869     self.ac_trlast = time.time()
1870     self.ac_mem = config.qdb_ac_mem
1871     self.ac_mem2 = 0
1872     self.start = self.ctime()
1873     self.port = port
1874     initVersion()
1875     SocketServer.ThreadingTCPServer.__init__(self, ("", port),
1876                                             QueryRequestHandler)
1877
1878     def process_jobs(self):
1879         self.jobsMutex.acquire()
1880         num_jobs = len(self.jobs)
1881         self.jobsMutex.release()
1882         for index in range(num_jobs):
1883             self.jobsMutex.acquire()
1884             if len(self.jobs):
1885                 job = self.jobs[0]
1886             else:
1887                 job = None
1888             self.jobsMutex.release()
1889             if not job:
1890                 continue
1891             try:
1892                 cid = job[0]
1893                 name = job[1]
1894                 if name == "setCache":
1895                     ncache = job[2]
1896                     debug('self.cache.statistic()'+'\n')
1897                     debug("*** neuer Cache ***", 2)
1898                     self.cache = ncache
1899             except:
1900                 import traceback
1901                 import StringIO
1902                 tb = StringIO.StringIO()
1903                 traceback.print_exc(file=tb)
1904                 tb.seek(0)
1905                 sys.stderr.write("\twhile doing job %s:\n" % 'job')
1906                 zeile = tb.readline()
1907                 while zeile != "":
1908                     sys.stderr.write("\t"+zeile)
1909                     zeile = tb.readline()
1910             self.jobsMutex.acquire()
1911             if len(self.jobs):
1912                 del self.jobs[0]
1913             self.jobsMutex.release()
1914
1915     def count_time(self, step):
1916         self.tcount = self.tcount + step
1917         if self.tcount > (60*30):
1918             self.tcount = 0
1919             if time.time() - self.tlast > (60*60):
1920                 self.tlast = time.time()

```

```

1921         debug('*** %s ***\n' % time.ctime(self.tlast), -5)
1922
1923 def check4autoclear(self):
1924     if self.ac_tclast <= self.tcount < (self.ac_tclast + self.ac_tcdist):
1925         return
1926     self.ac_tclast = self.tcount
1927     now = time.time()
1928     if now < (self.ac_trlast + self.ac_trdist):
1929         return
1930     self.ac_trlast = now
1931     try:
1932         mem = getVmSize()
1933         l = string.split(mem)
1934         zahl = l[0]
1935         einheit = l[1]
1936         mem_kb = int(zahl)
1937         if einheit in ["KB", "kB"]:
1938             pass
1939         elif einheit in ["MB", "mB"]:
1940             mem_kb = mem_kb * 1024
1941         else:
1942             raise Exception
1943         mem = mem_kb
1944     except:
1945         debug("*** autoclear failed: unable to get VmSize ***\n", -12)
1946         return
1947     mem2 = self.ac_mem2
1948     if mem2 < self.ac_mem:           # self.ac_mem nicht unterschreiten
1949         mem2 = self.ac_mem
1950     if mem2 >= mem:                 # mem noch unter dem Limit mem2?
1951         mem2 = self.ac_mem2 / 2     # mem2 verkleinern
1952         if mem2 < self.ac_mem:      # self.mem nicht unterschreiten
1953             self.ac_mem2 = self.ac_mem
1954         else:
1955             self.ac_mem2 = mem2
1956     return
1957     self.ac_mem2 = mem              # aktuellen Speicherbedarf als Limit
1958     qrhid = self.nextqrhid
1959     self.nextqrhid = qrhid + 1
1960     self.qrhFinishedMutex.acquire()
1961     self.qrhFinished[qrhid] = 0
1962     self.qrhFinishedMutex.release()
1963     self.infoMutex.acquire()
1964     s = "cache autoclear %s startet on %s" % ('qrhid', 'self.ctime()')
1965     self.info[qrhid] = s
1966     self.infoMutex.release()
1967     import threading
1968     t = threading.Thread(target = self.p_autoclear, args = (qrhid, ))
1969     t.start()
1970     debug("*** autoclear startet on %s ***\n" % self.ctime(), -5)
1971
1972 def p_autoclear(self, qrhid):
1973     self.cache.invalidate()
1974     self.infoMutex.acquire()
1975     s = "cache autoclear %s finished on %s" % ('qrhid', 'self.ctime()')
1976     self.info[qrhid] = s
1977     self.infoMutex.release()
1978     self.qrhFinishedMutex.acquire()
1979     self.qrhFinished[qrhid] = 1     # alter server-data
1980     self.qrhFinishedMutex.release()
1981

```

```

1982     def get_request(self):
1983         r = []
1984         while not self.stop and not r:
1985             r,w,e = select.select([self.socket], [], [], 4) # wait max 4 sec
1986             self.count_time(4)
1987             self.process_jobs()
1988             self.check4autoclear()
1989         if r:
1990             return self.socket.accept()
1991         else:
1992             return (None, None)
1993
1994     def verify_request(self, request, client_address):
1995         if request:
1996             return 1
1997         return 0
1998
1999     def finish_request(self, request, client_address):
2000         qrhid = self.nextqrhid
2001         self.nextqrhid = qrhid + 1
2002         self.qrhFinishedMutex.acquire()
2003         self.qrhFinished[qrhid] = 0
2004         self.qrhFinishedMutex.release()
2005         self.infoMutex.acquire()
2006         self.info[qrhid] = "client %s startet on %s" % ('qrhid', 'self.ctime()')
2007         self.infoMutex.release()
2008         dbconn = self.dbconn
2009         self.RequestHandlerClass(request, client_address, self, qrhid, dbconn)
2010
2011     def handle_error(self, conn, client_address):
2012         self.stop = 1 # called if RequestHandler throws an exception
2013                     # (according to docu, but that's wrong if threaded)
2014
2015     def ctime(self):
2016         return time.ctime(time.time())
2017
2018     def serve_until_shutdown(self):
2019         self.stop = 0
2020         self.tlast = time.time()
2021         debug('*** Service started on %s ***\n' % self.ctime(), -7)
2022         while not self.stop:
2023             self.handle_request()
2024             self.qrhFinishedMutex.acquire()
2025             self.del_info_where_finished()
2026             self.del_1_from_qrhfinished()
2027             self.qrhFinishedMutex.release()
2028         debug('*** Service stopped on %s ***\n' % self.ctime(), -7)
2029         time.sleep(0.05)
2030         self.qrhFinishedMutex.acquire()
2031         qrh_running = (0 in self.qrhFinished.values())
2032         self.qrhFinishedMutex.release()
2033         while qrh_running: # wait for termination of query threads
2034             time.sleep(2.0)
2035             self.qrhFinishedMutex.acquire()
2036             self.del_info_where_finished()
2037             self.del_1_from_qrhfinished()
2038             debug('waiting for childs %s\n' % self.qrhFinished.keys(), -6)
2039             qrh_running = (0 in self.qrhFinished.values())
2040             self.qrhFinishedMutex.release()
2041         debug('self.cache.statistic()'+'\n')
2042

```

```

2043 def del_info_where_finished(self):
2044     self.infoMutex.acquire()
2045     keys = self.info.keys()
2046     self.infoMutex.release()
2047     for cid in keys:
2048         self.infoMutex.acquire()
2049         if self.info.has_key(cid) \
2050             and self.qrhFinished.has_key(cid) \
2051             and self.qrhFinished[cid] == 1:
2052             del self.info[cid]
2053         self.infoMutex.release()
2054
2055 def del_1_from_qrhfinished(self):
2056     for key in self.qrhFinished.keys():
2057         if self.qrhFinished[key] == 1:
2058             del self.qrhFinished[key]
2059
2060 def close(self):
2061     self.socket.close()
2062     self.dbconn.close()
2063
2064 def p_addToStatistic(self, s):
2065     for name in s.keys():
2066         if self.zeit_in.has_key(name):
2067             bis_jetzt = self.zeit_in[name]
2068         else:
2069             bis_jetzt = 0.0
2070         try:
2071             bis_jetzt = bis_jetzt + s[name]
2072             self.zeit_in[name] = bis_jetzt
2073         except TypeError:
2074             debug("Fehler: zeit_in (Server) %s\n" %'(self.zeit_in,s)', -3)
2075
2076
2077 def debug(line, level = 8):
2078     if debug_level >= level:
2079         sys.stderr.write(line)
2080
2081 def reverse(s):
2082     m = len(s)-1
2083     r = ""
2084     for i in range(m+1):
2085         r = r + s[m-i]
2086     return r
2087
2088 def getVmSize():
2089     pid = os.getpid()
2090     pinfo = open("/proc/%s/status" %pid, "r")
2091     size = "unbekannt"
2092     if pinfo:
2093         line = pinfo.readline()
2094         while line != "":
2095             if line[:7] == "VmSize:":
2096                 size = string.strip(line[7:])
2097             line = pinfo.readline()
2098     pinfo.close()
2099     return size
2100
2101 def initVersion():
2102     global version
2103     version = {}

```

```

2104     if os.path.exists(config.databaseapi + "CVS/Entries"):
2105         entries = open(config.databaseapi + "CVS/Entries", "r")
2106         line = entries.readline()
2107         while line != "":
2108             d = string.split(line, '/')
2109             if len(d) >= 3 and d[1] in ["querydb.py", "config.py", "db.py"]:
2110                 version[d[1]] = d[2]
2111             line = entries.readline()
2112         entries.close()
2113
2114 if __name__ == '__main__':
2115     sys.stderr.write("Please use ../demo-qdb.py\n")
2116
2117

```

qclient.py

Der Datenbankclient.

```

1  #!/usr/bin/python
2
3  # qclient.py: Client zum querydb-Server
4
5  import secret
6  import cache
7  import config
8  import regiondb
9  from socket import *
10 import base64
11 import pickle
12 import string
13 import sys
14 import zlib
15
16 class Query: #{
17
18     def __init__(self, address = None):
19         if not regiondb.defaultDbconn:
20             sys.stderr.write("erst regiondb.setDefaultDbconn() aufrufen!\n")
21             raise Exception("erst regiondb.setDefaultDbconn() aufrufen!")
22         self.cache = cache.DefaultCache(self)
23         self.sock = None
24         self.param = {}
25         self.paramHash = ""
26         if not address:
27             for pport in config.qdb_ports:
28                 if self.connect((config.qdb_host, pport)):
29                     address = (config.qdb_host, pport)
30                     #self.close()
31                     break
32         else:
33             self.connect(address)
34         self.address = address
35         sys.stderr.write("Benutze Server " + 'address' + "\n")
36         self.korplist = None
37
38     def ok(self):
39         return self.address
40
41     def connect(self, address = None):
42         if self.sock:
43             self.close()

```

```

44     if not address:
45         address = self.address
46     if not address:
47         return 0
48     sock = socket(AF_INET, SOCK_STREAM)
49     try:
50         sock.connect(address)
51     except error, data:
52         return 0
53     frage = sock.recv(16384)
54     pos = string.find(frage, 'getValue(')
55     start = string.find(frage, '"', pos) + 1
56     ende = string.find(frage, '"', start)
57     if (start >= 0) and (ende >= 0) and (start <= ende):
58         key = frage[start:ende]
59     else:
60         sock.close()
61         return 0
62     sock.send(secret.getValue(key))
63     antwort = sock.recv(16384)
64     if string.rstrip(antwort) == "OK":
65         self.sock = sock
66         return 1
67     else:
68         sock.close()
69         return 0
70
71     # key aus Frage extrahieren, getValue aufrufen, Antwort verschicken
72
73 def close(self):
74     if self.sock:
75         self.sock.close()
76         self.sock = None
77
78 def p_checkNotInStr(self, badchars, str):
79     for c in badchars:
80         if c in str:
81             raise ValueError # gibt's dafuer eine besser passende Exception?
82
83 def p_updateParam(self, key, value):
84     self.param[key] = value
85     liste = self.param.values()
86     liste.sort()
87     self.paramHash = `liste`
88
89 def p_setcorpus(self, idlststr):
90     self.p_checkNotInStr(';\\n', idlststr)
91     if self.sock:
92         self.p_process_line("u:" + idlststr)
93     self.p_updateParam("corpus", "u:" + idlststr)
94
95 def setkorpus(self, qdb_type):
96     self.p_checkNotInStr(':\\n', qdb_type)
97     self.p_setcorpus(qdb_type)
98
99 def setwbkorpus(self):
100     self.p_setcorpus("wb")
101
102 def seterkwbkorpus(self):
103     self.p_setcorpus("wb+erkannt")
104

```

```

105 def seterkkorpus(self):
106     self.p_setcorpus("erkannt")
107
108 def p_setParam(self):
109     for line in self.param.values():
110         self.p_process_line(line)
111
112 def p_getAntwort(self):
113     antwort = self.sock.recv(16384)
114     ETB = "<DONE>\n" # auf Wunsch von Arno ein verstaedlicher String
115     linesrecved = string.count(antwort, "\n")
116     fertig = (string.find(antwort, ETB) >= 0)
117     while not fertig:
118         temp = self.sock.recv(16384)
119         if not temp: # connection closed?
120             raise Exception, "connection closed before ETB"
121         linesrecved = linesrecved + string.count(temp, "\n")
122         antwort = antwort + temp
123         if len(temp) > len(ETB):
124             fertig = (string.find(temp, ETB) >= 0) # schnell
125         else:
126             fertig = (string.find(antwort, ETB) >= 0) # langsam
127     return antwort[:string.find(antwort, ETB)]
128
129 def p_isCommand(self, line):
130     if line[:8] in ["shutdown"]:
131         return 1
132     if line[:6] in ["status"]:
133         return 1
134     if line[:5] in ["cache", "debug"]:
135         return 1
136     if line[:4] in ["info"]:
137         return 1
138     if line[:2] in ["u:"]:
139         return 1
140     return 0
141
142 def p_isUpdate(self, line):
143     if string.find(line,"setparse") >= 0:
144         return 1
145     else:
146         return 0
147
148 def p_process_line(self, line):
149     isUpdate = self.p_isUpdate(line)
150     if self.cache.has_key((self.paramHash, line)) \
151     and not self.p_isCommand(line): # Kommandos nicht cachen
152         antwort = self.cache[(self.paramHash, line)]
153     elif not self.sock: #{
154         self.connect()
155         if self.param:
156             self.p_setParam()
157         self.sock.send(line+"\n")
158         antwort = self.p_getAntwort()
159         self.close()
160     #}
161     else:
162         self.sock.send(line+"\n")
163         antwort = self.p_getAntwort()
164     if not isUpdate:
165         self.cache[(self.paramHash, line)] = antwort

```



```

166     else:
167         self.cache.invalidate()
168     return antwort
169
170 def wort(self, **where):
171     return self.p_execute('wort', where)
172
173 def lemma(self, **where):
174     return self.p_execute('grundform', where)
175
176 def satz(self, **where):
177     return self.p_execute('satz', where)
178
179 def segment(self, **where):
180     return self.p_execute('segment', where)
181
182 def p_execute(self, name, where):
183     try:
184         self.p_checkNotInStr('; \n', name)
185         q = "t:" + name + ";s:pyob;w:"
186         if len(where.keys()) < 1:
187             sys.stderr.write("qclient.Query.p_execute: where-Teil leer\n")
188         for key in where.keys():
189             value = where[key]
190             self.p_checkNotInStr('; \n=', key)
191             self.p_checkNotInStr('; \n' , value)
192             q = q + key + "=" + value + ","
193         obj64 = self.p_process_line(q[:-1]) # letztes Komma wegschneiden
194         if obj64[:8] == "<ERROR>\n":
195             sys.stderr.write(obj64)
196             raise Exception
197         #sys.stderr.write("obj64: ### BEGIN ###\n" + obj64 + "### END ###\n")
198         if obj64[:2] == "1\n":
199             objstr = base64.decodestring(obj64[2:])
200             objstr = zlib.decompress(objstr)
201         else:
202             objstr = obj64[2:]
203         return pickle.loads(objstr)
204     except:
205         return [] #kann passieren, wenn Satzzeichen dazwischenkommen :(
206
207 def query(self, line):
208     return (self.p_process_line(string.rstrip(line)), None)
209
210 def unpack(self, writefun, object, select):
211     apply(writefun, (object,))
212
213 #}
214
215 def main():
216     regiondb.setDefaultDbconn(db.Db())
217     qdb = None
218     try:
219         if len(sys.argv) > 1:
220             port = sys.argv[1]
221             qdb = Query((config.qdb_host, int(port)))
222         else:
223             qdb = Query()
224         while 1:
225             sys.stdout.write("->")
226             line = raw_input()

```

```

227         if not line:
228             break
229         #         if qdb.connect():
230             query, select = qdb.query(line)
231             qdb.unpack(sys.stdout.write, query, select)
232         qdb.close()
233         sys.exit()
234     except error, data:
235         print error, data
236         if qdb:
237             qdb.close()
238         sys.exit(1)
239
240 if __name__ == '__main__':
241     main()
242

```

cache.py

Ein Cache, zum Speichern auf die Platte.

```

1  #/usr/bin/python
2
3  # Modul Lexikon.cache
4
5  import sys
6  import thread
7
8  class Cache:
9
10     def has_key(self, key):
11         self.miss = self.miss + 1
12         return 0
13
14     def keys(self):
15         return []
16
17     def values(self):      # list of values
18         liste = []
19         for key in self.keys():
20             liste.append(self[key])
21         return liste
22
23     def items(self):      # list of (key, value) pairs
24         liste = []
25         for key in self.keys():
26             liste.append((key, self[key]))
27         return liste
28
29     def __getitem__(self, key):
30         self.hit = self.hit + 1
31         return None
32
33     def __setitem__(self, key, value):
34         self.write = self.write + 1
35         pass
36
37     def __delitem__(self, key):
38         self.dele = self.dele + 1
39         pass
40
41     def __init__(self, dummy):

```

```
42     self.initstat()
43
44     def initstat(self):
45         self.miss = 0
46         self.hit = 0
47         self.write = 0
48         self.dele = 0
49
50     def __repr__(self):
51         return '{}\n'
52
53     def __getstate__(self):
54         return self.items()
55
56     def __setstate__(self, state):
57         self.initstat()
58
59     def invalidate(self):
60         for key in self.keys():
61             del self[key]
62
63     def statistic(self):
64         total = self.hit+self.miss
65         if total > 0:
66             saved = 100*self.hit/total
67         else:
68             saved = 0
69         return {"miss" : self.miss,
70               "hit" : self.hit,
71               "reads saved in %" : saved,
72               "write" : self.write,
73               "del" : self.dele}
74
75     class NullCache(Cache):
76         pass
77
78     class FullCache(Cache):
79
80         def has_key(self, key):
81             retval = self.inhalt.has_key(key)
82             if not retval:
83                 self.miss = self.miss + 1
84             return retval
85
86         def keys(self):
87             Cache.keys(self)
88             return self.inhalt.keys()
89
90         def __getitem__(self, key):
91             Cache.__getitem__(self, key)
92             return self.inhalt[key]
93
94         def __setitem__(self, key, value):
95             Cache.__setitem__(self, key, value)
96             self.inhalt[key] = value
97
98         def __delitem__(self, key):
99             Cache.__delitem__(self, key)
100             del self.inhalt[key]
101
102     def __init__(self, basis):
```

```

103     Cache.__init__(self, basis)
104     self.inhalt = {}
105
106     def __setstate__(self, state):
107         Cache.__setstate__(self, [])
108         self.inhalt = {}
109         for (key, value) in state:
110             self[key] = value
111
112     def __repr__(self):
113 #         Cache.(self)
114         return `self.inhalt`
115
116 class MutexDefaultCache(FullCache):
117
118     def has_key(self, key):          # Warnung: Zustand kann sich aendern
119         self.mutex.acquire()
120         retval = FullCache.has_key(self, key)
121         self.mutex.release()
122         return retval
123
124     def keys(self):                 # Warnung: Zustand kann sich aendern
125         self.mutex.acquire()
126         retval = FullCache.keys(self)
127         self.mutex.release()
128         return retval
129
130     def items(self):               # list of (key, value) pairs
131         liste = []
132         for key in self.keys():
133             try:                   # sicherstellen, dass Paar noch da
134                 item = (key, self[key])
135                 liste.append(item)
136             except KeyError:
137                 pass
138         return liste
139
140     def __getitem__(self, key):
141         self.mutex.acquire()
142         keyerror = 0
143         try:
144             retval = FullCache.__getitem__(self, key)
145         except KeyError:
146             self.miss = self.miss + 1
147             keyerror = 1
148         self.mutex.release()
149         if keyerror:
150             raise KeyError
151         return retval
152
153     def __setitem__(self, key, value):
154         self.mutex.acquire()
155         FullCache.__setitem__(self, key, value)
156         self.mutex.release()
157
158     def __delitem__(self, key):
159         self.mutex.acquire()
160         FullCache.__delitem__(self, key)
161         self.mutex.release()
162
163     def __init__(self, basis):

```

```

164     FullCache.__init__(self, basis)
165     self.mutex = thread.allocate_lock()
166
167     def __setstate__(self, state):
168         FullCache.__setstate__(self, [])
169         self.mutex = thread.allocate_lock()
170         for (key, value) in state:
171             self[key] = value
172
173
174 class SkipCache(Cache):
175
176     def has_key(self, key):
177         Cache.has_key(self, key)
178         return self.inhalt.has_key(key)
179
180     def keys(self):
181         Cache.keys(self)
182         return self.inhalt.keys()
183
184     def __getitem__(self, key):
185         Cache.__getitem__(self, key)
186         return self.inhalt[key]
187
188     def __setitem__(self, key, value):
189         if len(self.keys()) > self.limit:           # 2do: Speicherbelegung pruefen
190             self.inhalt = {}
191         Cache.__setitem__(self, key, value)
192         self.inhalt[key] = value
193
194     def __delitem__(self, key):
195         Cache.__delitem__(self, key)
196         del self.inhalt[key]
197
198     def __init__(self, basis):
199         Cache.__init__(self, basis)
200         self.inhalt = {}
201
202     def __repr__(self):
203         # Cache.(self)
204         return `self.inhalt`
205
206
207 class OrderedCache(Cache): #{                               # abstrakte Klasse
208
209     def p_newStateFor(self, key, value):                   # ueberschreiben
210         return 0
211
212     def p_stateCmp(self, state1, state2):                  # ueberschreiben
213         return 0
214
215     def has_key(self, key):
216         Cache.has_key(self, key)
217         return self.inhalt.has_key(key)
218
219     def keys(self):
220         Cache.keys(self)
221         return self.inhalt.keys()
222
223     def __getitem__(self, key):
224         Cache.__getitem__(self, key)

```

```

225     value, index = self.inhalt[key]
226     state, keyv = self.heap[index]
227     if key != keyv:
228         sys.stderr.write("Fehler in DatabaseAPI.cache.OrderedCache\n")
229     state = self.p_newStateFor(state, key, value)
230     self.heap[index] = (state, key)
231     # 2do: Kosten anpassen und del self.heap[0] aufrufen, bis gering genug
232     # !!! erstmal hier nicht weiter machen, da SkipCache zuerst schreiben
233     return value
234
235     def __setitem__(self, key, value):
236         Cache.__setitem__(self, key, value)
237         # 2do
238         self.inhalt[key] = value
239
240     def __delitem__(self, key):
241         Cache.__delitem__(self, key)
242         # 2do
243         del self.inhalt[key]
244
245     def __init__(self, basis):
246         Cache.__init__(self, basis)
247         # 2do
248         self.inhalt = {}
249
250     def __repr__(self):
251     #         Cache.(self)
252         # 2do
253         return `self.inhalt`
254 #}
255
256
257 class Heap: #{
258
259     def __init__(self, cmpF, size):
260         self.cmpF = cmpF
261         self.data = size * [None]
262
263     #2do : setitem -> Reorganisation
264
265 #}
266
267
268 class DefaultCache(FullCache):
269     pass # 2do: Methoden def.
270
271
272 class Cached(DefaultCache): #{ # Cache durch Vererbung
273
274     def loaditem(self, key): # ueberschreiben
275         return None
276
277     def saveitem(self, key, value): # ueberschreiben
278         pass
279
280     def loadable(self, key): # ueberschreiben
281         return 0
282
283     def __init__(self, client):
284         DefaultCache.__init__(self, client)
285

```

```

286 def __getitem__(self, key):
287     if DefaultCache.has_key(self, key):
288         return DefaultCache.__getitem__(self, key)
289     value = self.loaditem(key)
290     DefaultCache.__setitem__(self, key, value)
291     return value
292
293 def __setitem__(self, key, value):
294     self.saveitem(key, value)
295     DefaultCache.__setitem__(self, key, value)
296
297 def haskey(self, key):
298     return DefaultCache.has_key(self, key) or \
299         self.loadable(key)
300 #}
301
302 if __name__ == '__main__':
303     mc = MutexDefaultCache(None)
304     mc['a'] = 'A'
305     mc['b'] = 'B'
306     print mc['a']
307     mc['c'] = 'C'
308     print mc['b']
309     print mc.statistic()
310
311

```

config.py

Konfigurationseinzelheiten.

```

1  #!/usr/bin/python
2
3  # Modul config.py
4  #
5  # Attribute:
6  #     dbname
7  #     hostname
8  #     user
9  #     tagger_de
10 #     tagger_en
11 #     wdateien
12
13 import getpass
14 import os
15 import popen2
16 import string
17 import sys
18
19 config = {
20
21 #
22 # Name und Typ der Datenbank
23 #
24
25 "dbname"      : {"koks@nulix"          : "mogell7b" ,
26                 "bkoch@nulix"        : "alternativ" ,
27                 "logotax@nulix"       : "neul7" ,
28                 "jowagner@nulix"      : "neul7" ,
29                 "ptschorn@nulix"      : "koks" ,
30                 "aerpenbe@nulix"      : "koks" ,
31                 "britta@inanna.surf2000.de" : "neu"

```

```

32         "koks@inanna.surf2000.de"           : "wb" ,
33         "koks@gorina"           : "neu17b" ,
34     },
35     "dbtype"      : { "koks@nulich"           : "mysql" ,
36                     "bkoch@nulich"          : "mysql" ,
37                     "logotax@nulich"         : "mysql" ,
38                     "jowagner@nulich"        : "mysql" ,
39                     "ptschorn@nulich"        : "mysql" ,
40                     "aerpenbe@nulich"        : "mysql" ,
41                     "koks@gorina"            : "mysql" ,
42                     "britta@inanna.surf2000.de" : "mysql" ,
43                     "koks@inanna.surf2000.de" : "mysql" ,
44     },
45     "dbuser"      : { "koks@nulich"           : "koks" ,
46                     "bkoch@nulich"          : "bkoch" ,
47                     "logotax@nulich"         : "koks" ,
48                     "jowagner@nulich"        : "koks" ,
49                     "ptschorn@nulich"        : "koks" ,
50                     "aerpenbe@nulich"        : "koks" ,
51                     "britta@inanna.surf2000.de" : "britta" ,
52                     "koks@inanna.surf2000.de" : "koks" ,
53                     "koks@gorina"            : "koks" ,
54     },
55     "dbpass"      : { "koks@nulich"           : "iKK410mn" ,
56                     "bkoch@nulich"          : "onK75Tdo" ,
57                     "logotax@nulich"         : "iKK410mn" ,
58                     "jowagner@nulich"        : "iKK410mn" ,
59                     "ptschorn@nulich"        : "iKK410mn" ,
60                     "aerpenbe@nulich"        : "iKK410mn" ,
61                     "britta@inanna.surf2000.de" : "iKK410mn" ,
62                     "koks@inanna.surf2000.de" : "iKK410mn" ,
63                     "koks@gorina"            : "iKK410mn" ,
64     },
65
66     #
67     # querydb Server, den der Client benutzen soll
68     #
69
70     "qdb_ports" : { "koks@nulich"           : [58597,45628,29295,55776,46581,55149] ,
71                   "bkoch@nulich"          : [58597,45628,29295,55776,46581,55149] ,
72                   "logotax@nulich"         : [55776,55149] ,
73                   "jowagner@nulich"        : [51111,45628,29295,55776,46581,55149] ,
74                   "ptschorn@nulich"        : [58597,45628,29295,55776,46581,55149] ,
75                   "aerpenbe@nulich"        : [58597,45628,29295,55776,46581,55149] ,
76                   "koks@gorina"            : [58597,45628,29295,55776,46581,55149] ,
77                   "britta@inanna.surf2000.de" : [58597,45628,29295,55776,46581,55149] ,
78                   "koks@inanna.surf2000.de" : [58597,45628,29295,55776,46581,55149] ,
79     },
80
81     "qdb_host"   : { "koks@nulich"           : "nulich.cl-ki.uni-osnabrueck.de" ,
82                   "bkoch@nulich"          : "nulich.cl-ki.uni-osnabrueck.de" ,
83                   "logotax@nulich"         : "nulich.cl-ki.uni-osnabrueck.de" ,
84                   "jowagner@nulich"        : "nulich.cl-ki.uni-osnabrueck.de" ,
85                   "ptschorn@nulich"        : "nulich.cl-ki.uni-osnabrueck.de" ,
86                   "aerpenbe@nulich"        : "nulich.cl-ki.uni-osnabrueck.de" ,
87                   "koks@gorina"            : "localhost" ,
88                   "britta@inanna.surf2000.de" : "localhost" ,
89                   "koks@inanna.surf2000.de" : "localhost" ,
90     },
91
92     #

```



```

93 # querydb Server cache.autoclear Schwelle
94 #
95
96 "qdb_ac_mem" : {"koks@nunix"      : 102102,
97               "bkoch@nunix"     : 102102,
98               "logotax@nunix"   : 102102,
99               "jowagner@nunix"  : 102102,
100              "ptschorn@nunix"   : 102102,
101              "aerpenbe@nunix"   : 102102,
102              "koks@gorina"      : 204204,
103              "britta@inanna.surf2000.de" : 102102,
104              "koks@inanna.surf2000.de" : 102102,
105              },
106
107 #
108 # Pfad zum Aligner
109 #
110
111 "align"      : {"koks@nunix"      : "/home/koks/align/align.sh" ,
112               "bkoch@nunix"     : "./align" ,
113               "jowagner@nunix"  : "/home/jowagner/bin/align" ,
114               "logotax@nunix"   : "/home/logotax/bin/align" ,
115               "aerpenbe@nunix"   : "/home/koks/align/align.sh" ,
116               "ptschorn@nunix"   : "./align" ,
117               "koks@gorina"      : "/home/koks/align/align.sh" ,
118               "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/align.sh" ,
119               "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/align.sh" ,
120              },
121
122 "al_jar"     : {"koks@nunix"      : "/home/koks/align/al.jar" ,
123               "bkoch@nunix"     : "/home/bkoch/koks/align/al.jar" ,
124               "jowagner@nunix"  : "/home/jowagner/align/al.jar" ,
125               "logotax@nunix"   : "/home/logotax/align/al.jar" ,
126               "aerpenbe@nunix"   : "./al.jar" ,
127               "ptschorn@nunix"   : "./al.jar" ,
128               "koks@gorina"      : "/home/koks/align/al.jar" ,
129               "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/al.jar" ,
130               "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/al.jar" ,
131              },
132
133 #
134 # Tagfolgenverzeichnis
135 #
136 "tags"      : {"koks@nunix"      : "/home/koks/align/tags" ,
137               "bkoch@nunix"     : "tags" ,
138               "jowagner@nunix"  : "/home/jowagner/align/tags" ,
139               "logotax@nunix"   : "/home/logotax/align/tags" ,
140               "aerpenbe@nunix"   : "./tags" ,
141               "ptschorn@nunix"   : "./tags" ,
142               "koks@gorina"      : "/home/koks/align/tags" ,
143               "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/tags" ,
144               "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/tags" ,
145              },
146
147 #
148 # DatabaseAPI-Verzeichnis
149 #
150 "databaseapi" : {"koks@nunix"      : "/home/koks/align/DatabaseAPI/" ,
151                "bkoch@nunix"     : "./DatabaseAPI/" ,
152                "jowagner@nunix"  : "/home/jowagner/align/DatabaseAPI/" ,
153                "logotax@nunix"   : "/home/logotax/align/DatabaseAPI/" ,

```

```

154     "aerpenbe@nunix"      : "./DatabaseAPI/" ,
155     "ptschorn@nunix"     : "./DatabaseAPI/" ,
156     "koks@gorina"        : "/home/koks/align/DatabaseAPI/" ,
157     "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/DatabaseAPI/" ,
158     "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/align/DatabaseAPI/" ,
159     },
160
161 #
162 # Pfad zum IMS Tree-Tagger
163 #
164
165 "tagger_de" : { "koks@nunix"          : "/home/koks/bin/tree-tagger-german" ,
166               "bkoch@nunix"         : "/home/bkoch/koks/bin/tree-tagger-german" ,
167               "jowagner@nunix"      : "/home/jowagner/bin/tree-tagger-german" ,
168               "logotax@nunix"       : "/home/logotax/bin/tree-tagger-german" ,
169               "aerpenbe@nunix"      : "/home/aerpenbe/bin/tree-tagger-german" ,
170               "ptschorn@nunix"      : "/home/koks/bin/tree-tagger-german" ,
171               "koks@gorina"         : "/home/koks/bin/tree-tagger-german" ,
172               "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-german",
173               "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-german" ,
174               },
175 "tagger_en" : { "koks@nunix"          : "/home/koks/bin/tree-tagger-english" ,
176               "bkoch@nunix"         : "/home/bkoch/koks/bin/tree-tagger-english" ,
177               "jowagner@nunix"      : "/home/jowagner/bin/tree-tagger-english" ,
178               "logotax@nunix"       : "/home/logotax/bin/tree-tagger-english" ,
179               "aerpenbe@nunix"      : "/home/koks/bin/tree-tagger-english" ,
180               "ptschorn@nunix"      : "/home/koks/bin/tree-tagger-english" ,
181               "koks@gorina"         : "/home/koks/bin/tree-tagger-english" ,
182               "britta@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-english",
183               "koks@inanna.surf2000.de" : "/home/data/studium/projekt/cvs/bin/tree-tagger-english" ,
184               },
185
186 #
187 # Groesse des zu verwendenen Woerterbuchs, beeinflusst wdateien
188 #
189
190 "wbsize"      : { "koks@nunix"          : "big",
191               "bkoch@nunix"         : "big",
192               "jowagner@nunix"      : "small",
193               "logotax@nunix"       : "small",
194               "aerpenbe@nunix"      : "small",
195               "ptschorn@nunix"      : "small",
196               "koks@gorina"         : "tiny",
197               "britta@inanna.surf2000.de" : "big" ,
198               "koks@inanna.surf2000.de" : "big" },
199
200 #Ids der Woerterbuecher
201 "wbids": { "koks@nunix"          : "(0,1,2,3,4)",
202           "bkoch@nunix"         : "(1,2,1225,1226,1227)",
203           "jowagner@nunix"      : "(0,1,2,3,4)",
204           "logotax@nunix"       : "(0,1,2,3,4)",
205           "ptschorn@nunix"      : "(1,2,3,1160,1161)",
206           "aerpenbe@nunix"      : "(1,2,3,1160,1161)",
207           "koks@gorina"         : "(0,1,2,3,4)",
208           "britta@inanna.surf2000.de" : "(0, 1, 2, 3, 4)",
209           "koks@inanna.surf2000.de" : "(0, 1, 2, 3, 4)" },
210
211 "goodcorpora": { "koks@nunix"          : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32,
212               "bkoch@nunix"         : "(7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24
213               "logotax@nunix"       : "(3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
214               "jowagner@nunix"      : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32,

```

```

215         "ptschorn@nunix"      : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32, 33, 34)",
216         "aerpenbe@nunix"     : "(5, 6, 9, 10, 11, 13, 14, 15, 17, 19, 20, 22, 24, 27, 28, 29, 30, 32, 33, 34)",
217         "koks@gorina"        : "(10)",
218         "britta@inanna.surf2000.de" : "(106, 107, 108, 109, 110, 111,112,113,114)",
219         "koks@inanna.surf2000.de" : "(106, 107, 108, 109, 110, 111,112,113,114)",
220     }
221 }
222
223 #
224 # --- ab hier braucht i.d.R. nichts geaendert werden ---
225 #
226
227 lang2id = {"de": '1', "en": '2'} # Bedeutung der SprachIDs
228
229 p_sys = sys.path[0]
230 if p_sys[:1] == "/":          # Pfad absolut?
231     p_cwd2 = p_sys           # Ja -> uebernehmen
232 else:
233     p_cwd = os.getcwd()      # Nein -> hoffentlich akt. Verz. noch unveraendert
234     os.chdir(p_cwd + "%s" %(sys.path[0]))
235     p_cwd2 = os.getcwd()
236     os.chdir(p_cwd)
237 is_in_bin = 0
238 if p_cwd2[-4:] == "/bin":
239     is_in_bin = 1
240 if p_cwd2[-5:] == "/bin/":
241     is_in_bin = 1
242
243 if is_in_bin and os.environ.has_key("KOKS_BIN_USER"):
244     user = os.environ["KOKS_BIN_USER"]
245 elif os.environ.has_key("KOKS_USER"):
246     user = os.environ["KOKS_USER"]
247 else:
248     user = getpass.getuser()
249 r,w=popen2.popen2("hostname")
250 w.close()
251 zeile=r.readline()
252 hostname = string.strip(zeile)
253 while zeile != "":
254     zeile=r.readline()
255 r.close()
256
257 # Folgende Variablen werden beim Import dieses Modules gesetzt:
258 dbname = config["dbname"][user + "@" + hostname]
259 dbtype = config["dbtype"][user + "@" + hostname]
260 dbpass = config["dbpass"][user + "@" + hostname]
261 dbuser = config["dbuser"][user + "@" + hostname]
262 qdb_ports = config["qdb_ports"][user + "@" + hostname]
263 qdb_host = config["qdb_host"][user + "@" + hostname]
264 qdb_ac_mem = config["qdb_ac_mem"][user + "@" + hostname]
265 align = config["align"][user + "@" + hostname]
266 al_jar = config["al_jar"][user + "@" + hostname]
267 tags = config["tags"][user + "@" + hostname]
268 databaseapi = config["databaseapi"][user + "@" + hostname]
269 tagger_de = config["tagger_de"][user + "@" + hostname]
270 tagger_en = config["tagger_en"][user + "@" + hostname]
271
272 wbsize = config["wbsize"][user + "@" + hostname]
273 wbids = config["wbids"][user + "@" + hostname]
274 goodcorpora = config["goodcorpora"][user + "@" + hostname]
275

```

```

276 if wbsize == "tiny": #{
277     wbdateien = [ {"de" : "../wb/wbtag/wb2/sk5000de.tag",
278                   "en" : "../wb/wbtag/wb2/sk5000en.tag"} ]
279 #}
280 elif wbsize == "small": #{
281     wbdateien = [ {"de" : "../wb/wbtag/wb2/sk20000de.tag",
282                   "en" : "../wb/wbtag/wb2/sk20000en.tag"} ]
283 #}
284 elif wbsize == "big": #{
285     wbdateien = [ {"de" : "../wb/wbtag/wb2/KOKSdt_eng2ger.tag",
286                   "en" : "../wb/wbtag/wb2/KOKSeng_eng2ger.tag"} ]
287 #}
288 elif wbsize == "huge": #{
289     wbdateien = [ {"de" : "../wb/wbtag/wb2/KOKSdt_eng2ger.tag",
290                   "en" : "../wb/wbtag/wb2/KOKSeng_eng2ger.tag"} ,
291                   {"de" : "../wb/wbtag/wb3/KOKSdt_German.tag",
292                   "en" : "../wb/wbtag/wb3/KOKSeng_German.tag"} ,
293                   {"de" : "../wb/wbtag/wb1/KOKSdt_ger-eng.tag",
294                   "en" : "../wb/wbtag/wb1/KOKSeng_ger-eng.tag"} ]
295 else: #{
296     sys.stderr.write("Fehler in config.py: wbsize falsch konfiguriert\n")
297     wbdateien = []
298 #}
299
300 def test():
301     usersAndHosts = {}
302     for key in config.keys():                # Benutzer sammeln
303         for userAndHost in config[key].keys():
304             usersAndHosts[userAndHost] = 1
305     for key in config.keys():                # Vollstaendigkeit pruefen
306         for userAndHost in usersAndHosts.keys():
307             if not config[key].has_key(userAndHost):
308                 print "In", key, "fehlt", userAndHost
309
310 if __name__ == '__main__':
311     test()

```

db.py

Eine Hilfsklasse für Datenbankzugriff.

```

1  #!/usr/bin/python
2
3  # Modul DatabaseAPI.db
4
5  #TODO: nur postgres!!!
6  #verkapselt Datenbank-Krams
7
8  #import pg
9  import config
10 import sys
11 import thread
12 import types
13
14 p_dbconn_count = 0
15
16 class Db:
17
18     def __init__(self, dbtype=config.dbtype, dbname=config.dbname):
19         self.initDb(dbtype, dbname)
20
21     def initDb(self, dbtype, dbname):

```

```

22     global p_dbconn_count
23     p_dbconn_count = p_dbconn_count + 1
24     if p_dbconn_count > 10:
25         sys.stderr.write("Warnung: %s db.Db Objekte\n" %p_dbconn_count)
26     if p_dbconn_count > 100:
27         sys.stderr.write("Das sind zu viele. stack trace folgt.\n")
28         import traceback
29         import StringIO
30         tb = StringIO.StringIO()
31         traceback.print_stack(file=tb)
32         tb.seek(0)
33         sys.stderr.write("in %s:\n" %sys.argv[0])
34         zeile = tb.readline()
35         while zeile != "":
36             sys.stderr.write("\t"+zeile)
37             zeile = tb.readline()
38     self.dbtype = dbtype
39     self.dbname = dbname
40     if self.dbtype == "pg":
41         import pg
42         self.dbconn = pg.connect(dbname)
43         self.api = 0
44     elif self.dbtype == "mysql":
45         import MySQLdb
46         self.dbconn = MySQLdb.connect(db=dbname, user=config.dbuser, passwd=config.dbpass)
47         self.api = 1
48         #         elif self.dbtype = "oracle":
49         #             import DCOracle
50         #             self.dbconn = DCOracle.Connect("westlotto/fee@q")
51         #             self.api = 1
52     else:
53         sys.stderr.write("dbtype " + `dbtype` + " unknown\n")
54     self.mutex = thread.allocate_lock()
55
56 def getMaxIndex(self, spalte, tabelle):
57     # 2do: what if two threads append items at the same time?
58     idx = self.executelist("select max(%s) from %s" %(spalte, tabelle))
59     if idx and idx[0] and idx[0][0]:
60         idx = idx[0][0]
61     else:
62         idx = 0L
63     return self.incr(idx)
64
65 def __getstate__(self):
66     return (self.dbtype, self.dbname)
67
68 def __setstate__(self, state):
69     self.initDb(state[0], state[1])
70
71 def incr(self, idx):
72     return unknown2string(long(idx) + 1)
73
74 def executedict(self, query):
75     self.mutex.acquire()
76     #sys.stderr.write("Db.executedict('+`query`+')\n")
77     if self.api:
78         self.cursor = self.dbconn.cursor()
79         self.cursor.execute(query)
80         try:
81             rows = self.cursor.fetchall()
82             colnames = []

```

```

83         result = []
84         for col in cursor.description():
85             colnames.append(col[0])
86         for row in rows:
87             dict = {}
88             i = 0
89             for col in row:
90                 dict[colnames[i]] = col
91                 i = i + 1
92             result.append(dict)
93         self.mutex.release()
94         return result
95     except Error, info:
96         print info
97     else:
98         self.mutex.release()
99         raise Exception
100 else:
101     self.results = self.dbconn.query(query)
102     if self.results:
103         retval = self.results.dictresult()
104         self.mutex.release()
105         return retval
106 self.mutex.release()
107 return None
108
109 def executelist(self, query):
110     self.mutex.acquire()
111     #sys.stderr.write("Db.executelist('+`query`+')\n")
112     if self.api:
113         self.cursor = self.dbconn.cursor()
114         self.cursor.execute(query)
115         try:
116             retval = self.cursor.fetchall()
117             self.mutex.release()
118             return retval
119         except Error, info:
120             print info
121         else:
122             self.mutex.release()
123             raise Exception
124     else:
125         self.results = self.dbconn.query(query)
126         if self.results:
127             retval = self.results.getresult()
128             self.mutex.release()
129             return retval
130     self.mutex.release()
131     return None
132
133 def update(self, query):
134     self.mutex.acquire()
135     #sys.stderr.write("Db.update('+`query`+')\n")
136     if self.api:
137         self.cursor = self.dbconn.cursor()
138         self.cursor.execute(query)
139     else:
140         self.dbconn.query(query)
141     self.mutex.release()
142
143 def close(self):

```

```

144     self.mutex.acquire()
145     if self.dbconn:
146         self.dbconn.close()
147         global p_dbconn_count
148         p_dbconn_count = p_dbconn_count - 1
149         if p_dbconn_count < 0:
150             sys.stderr.write("Fehler: db.p_dbconn_count = ")
151             sys.stderr.write("%s\n" %p_dbconn_count)
152     else:
153         sys.stderr.write("Db: None.close()\n")
154     self.dbconn = None        # we should test for it above
155     self.mutex.release()
156
157     def commit(self):
158         pass
159
160     def reset(self):
161         self.mutex.acquire()
162         if self.api:
163             self.dbconn.rollback()
164         else:
165             self.dbconn.reset()
166         self.mutex.release()
167
168
169     def incr(idx):
170         return unknown2string(long(idx) + 1)
171
172     def unknown2string(unknown):
173         if type(unknown) == types.StringType:
174             return unknown
175         if type(unknown) == types.LongType:
176             return `unknown`[:-1]
177         if type(unknown) == types.IntType:
178             return `unknown`
179         raise TypeError
180

```

secret.py

Ein Klasse zum Verschlüsseln von Kommunikation.

```

1  #!/usr/bin/python
2
3  # Modul secret.py
4
5  import base64
6  import math
7  import md5
8  import random
9  import string
10 import sys
11 import time
12
13 def getKey(): #{
14     global salt
15     m = md5.new()
16     m.update(salt)
17     m.update(randomString(128))
18     m.update(`time.time()`)
19     return string.rstrip(base64.encodestring(m.digest()))
20 #}

```

```

21
22 def getValue(key): #{
23     global salt
24     global secret
25     m = md5.new()
26     m.update(salt)
27     m.update(secret)
28     m.update(key)
29     return string.rstrip(base64.encodestring(m.digest()))
30
31 def valueOK(key, value): #{
32     return value == getValue(key)
33 #}
34
35
36 # -- folgende Funktionen und Variablen sind privat --
37
38 salt = "u/v8iI5IB7V0oojnhtLsI7R4a/1GCLqE"
39 secret = "Yi2UV9C648hByNkVyWTcIfWLDsIQIDY2"
40
41 def randomString(bits): #{
42     zeichen = bits * math.log(2) / math.log(62) # base64 ohne /+
43     tripel = zeichen / 4.0
44     retval = ""
45     while tripel > 0.0: #{
46         s = ""
47         for i in range(3):
48             char = random.randint(0,255);
49             s = s + chr(char)
50         id = string.rstrip(base64.encodestring(s))
51         if id == string.replace(id, "/", "") \
52         and id == string.replace(id, "+", ""):
53             tripel = tripel - 1
54             retval = retval + id
55     #}
56     return retval
57 #}
58
59 def main():
60     if len(sys.argv) > 1:
61         key = sys.argv[1]
62         print getValue(key)
63     else:
64         print "Usage: python secret.py <KEY>"
65
66 if __name__ == '__main__':
67     main()

```

demo-qdb.py

Kommandozeilenaufruf des Servers.

```

1 #!/usr/bin/python
2
3 from DatabaseAPI import querydb, qclient, config, regiondb, db
4 import readline # provide elaborate line editing in raw_input()
5 import socket
6 import string
7 import sys
8 import types
9
10 def main():

```



```

11 if len(sys.argv) <= 1:
12     sys.stderr.write("Usage: -client [ <port-key> ] | -server [ <port-key> ]\n")
13     sys.stderr.write("\t%s\t%s\n" %("key","port"))
14     for i in range(len(config.qdb_ports)):
15         sys.stderr.write("\t%s\t%s\n" %(i, config.qdb_ports[i]))
16     sys.exit(1)
17 regiondb.setDefaultDbconn(db.Db())
18 qdb = querydb.Query(None, 8)          # default: verbose
19 try:
20     if "-q" in sys.argv:              # quiet
21         qdb.close()
22         qdb = querydb.Query(None, 1)
23         sys.argv.remove("-q")
24     if len(sys.argv) > 1:
25         port = sys.argv[1]
26         if string.strip(port)[:2] == '-c':
27             qdb.close()
28             if len(sys.argv) > 2:
29                 i = int(sys.argv[2])
30                 port = config.qdb_ports[i]
31                 qdb=qclient.Query('',port)
32             else:
33                 qdb=qclient.Query()
34         else:
35             if string.strip(port)[:2] == '-s':
36                 if len(sys.argv) > 2:
37                     i = int(sys.argv[2])
38                 else:
39                     i = 1
40                 port = config.qdb_ports[i]
41                 sys.stderr.write("Server auf Port "+'port'+"\n")
42                 qdb.close()
43                 server=querydb.QueryServer(int(port))
44                 server.serve_until_shutdown()
45                 server.close()
46                 sys.exit()
47     while 1:
48         sys.stdout.write("->")
49         line = raw_input()
50         if not line:
51             break
52         query, select = qdb.query(line)
53         if type(qdb.unpack) == types.DictType:
54             qdb.unpack[select](sys.stdout.write, query)
55         else:
56             qdb.unpack(sys.stdout.write, query, select)
57     qdb.close()
58     sys.exit()
59 except socket.error, data:
60     print socket.error, data
61     qdb.close()
62     sys.exit()
63
64 if __name__ == '__main__':
65     main()
66
67

```

tagger.py

Wrapperklasse um den Tagger.

```

1  #!/usr/bin/python
2
3  # Modul tagger
4
5  from DatabaseAPI import config, cache
6  from Region         import absatz
7  import popen2
8  import string
9  import sys
10
11 tagcache = cache.DefaultCache('tagger')      # modulglobaler Cache
12
13 def tagline(line, lang):
14     if tagcache.has_key((lang, line)):
15         return tagcache[(lang, line)]
16     r,w=popen2.popen2(getattr(config, "tagger_" + lang) + " 2>/dev/null")
17     w.write(line)
18     w.write("\n<DOKENDE>\n*\n")
19     w.close()
20     getaggt=r.read()
21     r.close()
22     if getaggt[-1] == "\n":
23         getaggt = getaggt[:-1]      # letztes Newline entfernen
24     else:
25         sys.stderr.write("Warnung: letzte Tagzeile ohne Newline\n")
26     absaetze = string.split(getaggt, "<ABSATZ>\n")
27     zeilenliste = string.split(absaetze[0], "\n")
28     reg = absatz.AbsatzVonZeilenliste(zeilenliste[:-1])
29     tagcache[(lang, line)] = reg
30     return reg
31

```

O.3 Korpora und die Datenbank

O.3.1 korpus2db.py

Dieses Skript kann zum Eintragen von Korpora benutzt werden, schöner ist aber preproc.py (s. Kapitel K.2).

```

1  #!/usr/bin/python
2  #schreibt einen Korpus in die Datenbank
3  #TODO: Hacks weg - also mit mehr Tagsets umgehen koennen
4
5  import getopt
6  import string
7  import sys
8  import types
9
10 from DatabaseAPI import config, db, cache
11
12 class Korpus:
13
14     def __init__(self):
15         self.dbconn = db.Db()
16         self.word_idx = self.dbconn.getMaxIndex("tokenid", "tokens")
17         self.sent_idx = self.dbconn.getMaxIndex("satzid", "saetze")
18         self.map_idx = self.dbconn.getMaxIndex("mapid", "map_s_w")
19         self.segment_idx = self.dbconn.getMaxIndex("segmentnr", "map_s_w")
20         self.autor_idx = self.dbconn.getMaxIndex("autorid", "autoren")
21         self.lemma_idx = self.dbconn.getMaxIndex("grundformid", "grundformen")
22         self.trans_table = string.maketrans("\'\'\'", "###")

```

```

23     self.cache = cache.DefaultCache(self)
24
25 def txt2db(self, herkunft, tpl1, tpl2):
26     segstart = self.segment_idx
27     for tpl in [tpl1, tpl2]:
28         file = tpl[0]
29         lang = tpl[1]
30         langnr = config.lang2id[lang]
31         tagsetnm = tpl[2] + "_" + lang
32         print "entering %s into db..." %file
33         input = open(file, "r")
34         lines = string.split(input.read(), "<DOKENDE>")
35         segs = string.split(lines[0], "<segmentgrenze>")
36         self.tag_idx = self.dbconn.getMaxIndex("tagid", tagsetnm)
37         tagset = self.dbconn.executelist("select name, tagsetid from tagsets where sprachenid = %s and name
38         if tagset:
39             tagset = tagset[0]
40         self.entersegmente(segs, self.sent_idx, herkunft, langnr, tagset)
41         input.close()
42         self.segment_idx = segstart
43     self.dbconn.commit()
44
45 def entersegmente(self, segmente, sentstart, herkunft, sprache, tagset):
46     next = self.dbconn.incr(sentstart)
47     prev = -1
48     for seg_line in segmente:
49         for sent in string.split(seg_line, "<SATZ>"):
50             sent = string.strip(sent)
51             self.entersentence(sent, prev, next, herkunft, sprache, tagset)
52             prev = self.dbconn.incr(prev)
53             if prev == 0:
54                 prev = db.unknown2string(long(self.sent_idx) - 1)
55                 next = self.dbconn.incr(next)
56             self.segment_idx = self.dbconn.incr(self.segment_idx)
57     self.dbconn.update("update saetze set succ = -1 where satzid = %s" % (db.unknown2string(long(self.sent_idx)
58
59 def entersentence(self, sentence, prev, next, herkunft, sprache, tagset):
60     #print sentence
61     self.dbconn.update("insert into saetze(satzid, prev, succ, sprachenid, herkunftsid) values(%s, %s, %s, %s, %s)"
62     koks_tag = 0
63     for word in string.split(sentence, "\n"):
64         word = string.strip(word)
65         if word == "<KOKS>":
66             if string.find(sentence, "</KOKS>") >= 0:
67                 koks_tag = 1
68             else:
69                 break # Schleife verlassen (sent_idx nur noch zu erhoehen)
70         if koks_tag:
71             if word == "</KOKS>":
72                 koks_tag = 0
73             continue
74         self.enterword(string.translate(word, self.trans_table), tagset)
75     self.sent_idx = self.dbconn.incr(self.sent_idx)
76
77 def enterword(self, word, tagset):
78     #print `word`
79     if len(string.split(word, "\t")) == 3:
80         ckey = word
81         if self.cache.has_key(ckey):
82             wort_idx = self.cache[ckey]
83     else:

```

```

84     word = string.replace(word, '\\\\', '/')
85     wort, tag, grundform = string.split(word, "\t")
86     #print wort, tag, grundform
87     tagset_name = tagset[0]
88     tagset_idx = db.unknown2string(tagset[1])
89     tag_idx = self.enterunique(tag, tagset_name, "name", "tagid", "tag_idx")
90     gf_idx = self.enterunique_complex(grundform, "grundformen", "grundformid",
91     "where name = '%s' and tagid = %s and tagsetid = %s" %(grundform, tag_idx, tagset_idx),
92     "grundformid, name, reverse, tagid, tagsetid",
93     "%s, '%s', '%s', %s, %s" %(self.lemma_idx, grundform, self.p_reverse(grundform),
94     tag_idx, tagset_idx), "lemma_idx")
95     wort_idx = self.enterunique_complex(wort, "tokens", "tokenid",
96     "where name = '%s' and grundformid = %s and tagid = %s and tagsetid = %s" %(wort,
97     gf_idx, tag_idx, tagset_idx),
98     "tokenid, name, reverse, grundformid, tagid, tagsetid",
99     "%s, '%s', '%s', %s, %s, %s" %(self.word_idx, wort, self.p_reverse(wort),
100     gf_idx, tag_idx, tagset_idx), "word_idx")
101     self.cache[ckey] = wort_idx
102     self.dbconn.update("insert into map_s_w (mapid, tokenid, satzid, segmentnr) values(%s, %s, %s, %s)"
103     self.map_idx = self.dbconn.incr(self.map_idx)
104
105 def p_reverse(self, word):
106     str = ""
107     for l in word:
108         str = l + str
109     return str
110
111 def enterunique(self, unique, table, column, idxcolumn, max_idx):
112     return self.enterunique_complex(unique, table, idxcolumn, "where %s = '%s'" %(column, unique), "%s, %s")
113
114 def enterunique_complex(self, unique, table, idxcolumn, where, columns, values, max_idx):
115     idx = self.dbconn.executelist("select %s from %s %s" %(idxcolumn, table, where))
116     if idx:
117         idx = idx[0][0]
118     else:
119         self.dbconn.update("insert into %s (%s) values(%s)" %(table, columns, values))
120         idx = long(getattr(self, max_idx))
121         setattr(self, max_idx, db.unknown2string(idx + 1)) #unschoen, aber pass by ref ist halt nicht...
122     return db.unknown2string(idx)
123
124 def enterquelle(self, name, verlag, bemerkungen): #TODO schon drin???
125     qid = self.dbconn.getMaxIndex("quellenid", "quellen")
126     self.dbconn.update("insert into quellen(quellenid, name, verlag, bemerkungen) values(%s, '%s', '%s', '%s')")
127     return qid
128
129 def enterherkunft(self, name, autor, jahr, quellenid):
130     hid = self.dbconn.getMaxIndex("herkunftsaid", "herkunft")
131     aid = self.enterunique(autor, "autoren", "name", "autorid", "autor_idx")
132     self.dbconn.update("insert into herkunft(herkunftsaid, name, autorid, jahr, quellenid) values(%s, '%s', '%s', '%s', '%s')")
133     return hid
134
135 def getinfo(self, file, infodict):
136     print "Bitte machen Sie Angaben zu %s:" % file
137     for inf in infodict.keys():
138         if not infodict[inf]:
139             print inf, '?'
140             infodict[inf] = sys.stdin.readline()[:-1]
141     qid = self.enterquelle(infodict['Quelle'], infodict['Verlag'], infodict['Bemerkungen'])
142     hid = self.enterherkunft(infodict['Textname'], infodict['Autor'], infodict['Jahr'], qid)
143     sid = self.choose("sprachen", 'name', 'sprachenid', 'Welche Sprache hat die Datei %s?' % file)
144     return (hid, sid, infodict)

```

```

145
146 def choose(self, table, col, id, question):
147     langlist = self.dbconn.executelist("select %s, %s from %s" %(id, col, table))
148     nr = -1
149     while nr < 0 or not nr in map(lambda x: x[0], langlist):
150         print question
151         for row in langlist:
152             print row[0], row[1]
153         nr = sys.stdin.readline()[::-1]
154         try:
155             nr = int(nr)
156         except:
157             nr = -1
158     return nr
159
160 def close(self):
161     self.dbconn.close()
162
163
164 def main():
165     usage = """%s [-e <korpus1> <korpus2>] [-h]
166         -e: enter aligned korpora
167         -h: print this help
168         the options cannot be combined!!!
169         """ %sys.argv[0]
170     try:
171         (options, args) = getopt.getopt(sys.argv[1:], "eh")
172     except:
173         print usage
174         sys.exit()
175
176     if len(args) < 1 or len(options) > 1:
177         print usage
178         sys.exit()
179
180     korpus = Korpus()
181     if options[0][0] == '-e':
182         infodict = {'Textname': '', 'Autor': '', 'Jahr': '', 'Quelle': '', 'Verlag': '', 'Bemerkungen': ''}
183         (hid, sid1, infodict) = korpus.getinfo(args[0], infodict)
184         sid2 = 3 - sid1 #Hack! Sprachen entweder 1 oder 2...
185         print "entering texts into db"
186         korpus.txt2db(hid, (args[0], sid1), (args[1], sid2))
187     else:
188         print usage
189     korpus.close()
190
191 if __name__ == '__main__':
192     main()
193
194
195
196

```

O.3.2 delkorpus.py

Mit diesem Skript kann man die Korpora auflisten, bzw. einzelne löschen.

```

1 #!/usr/bin/python
2
3 import sys
4 from DatabaseAPI import db

```

```

5
6 def listkorporus(dbc):
7     korpora = dbc.executelist("select h.herkunftsid, h.name, q.name, q.bemerkungen from herkunft h, quellen q")
8     print "ID\tPath\tName\tNotes\n"
9     print 72 * "-"
10    for korpus in korpora:
11        print "%s\t%s\t%s\t%s\n" % (db.unknown2string(korpus[0]), korpus[1], korpus[2], korpus[3])
12    print 72 * "-"
13
14 def sel(dbc, string):
15     list = dbc.executelist(string)
16     # print string, list
17     if not list or not list[0] or not list[0][0]:
18         return -1
19     return int(db.unknown2string(list[0][0]))
20
21 def delkorporus(dbc, id):
22     answer = raw_input("really delete korpus " + id + " ??? (y/n) ")
23     if not answer in ["y", "Y"]:
24         return
25     qid = dbc.executelist("select quellenid from herkunft where herkunftsid = %s" %id)
26     if not qid:
27         print "korpus doesn't exist!!!"
28         return
29     minsatz = sel(dbc, "select min(satzid) from saetze where herkunftsid = %s" %id)
30     print "minimal sentence id %s" %minsatz
31     maxsatz = sel(dbc, "select max(satzid) from saetze where herkunftsid = %s" %id)
32     print "maximal sentence id %s" %maxsatz
33     minseg = sel(dbc, "select min(mapid) from map_s_w where satzid >= %s and satzid < %s" %(minsatz, maxsatz))
34     print "minimal segment id %s" %minseg
35     maxseg = sel(dbc, "select max(mapid) from map_s_w where satzid >= %s and satzid < %s" %(minsatz, maxsatz))
36     print "maximal segment id %s" %maxseg
37     if maxsatz == -1 or minsatz == -1 or maxseg == -1 or minseg == -1:
38         print "problem getting minimum / maximum values for deletion...quitting!"
39         return
40     dbc.update("delete from saetze where satzid between %s and %s" %(minsatz, maxsatz))
41     dbc.update("delete from map_s_w where mapid between %s and %s" %(minseg, maxseg))
42     dbc.update("delete from herkunft where herkunftsid = %s" %id)
43     if len(qid) == 1:
44         dbc.update("delete from quellen where quellenid = %s" %db.unknown2string(qid[0][0]))
45
46 def main():
47     if len(sys.argv) < 2:
48         print "Listing korpora..."
49         listkorporus(db.Db())
50         print sys.argv[0], "<korpus_id> to delete the wanted corpus"
51         sys.exit(0)
52     else:
53         delkorporus(db.Db(), sys.argv[1])
54         sys.exit(0)
55
56 if __name__ == '__main__':
57     main()

```

Anhang P

Demo-Code

P.1 Code für die Demo-Application

P.1.1 Java-Applet KoksAppe.java

```
1 package koks;
2
3 import java.applet.*;
4 import java.awt.*;
5 import java.awt.event.*;
6 import java.net.HttpURLConnection ;
7 import java.net.URL ;
8 import java.io.IOException;
9 import java.net.*;
10
11 /**
12  * Applet
13  * <P>
14  * @author Norman Kummer
15  * @version 0.2 - 15.06.2001
16  *
17  * @author Arno Erpenbeck
18  * @version 0.3 - 21.06.2001
19  *
20  * @author Norman Kummer
21  * @version 0.4 23.08.2001 (english)
22  *
23  * @author Arno Erpenbeck
24  * @version 0.4.1 - 23.08.2001
25  *
26  * @author Arno Erpenbeck
27  * @version 0.4.2 - 08.10.2001
28  */
29 public class KoksAppe extends Applet {
30
31 //Klasse, welche die Oberfläche abbildet
32
33     private static final String VERSION = "0.4.2";
34     //private static final String BASE_URL =
35 "http://nunix.cl-ki.uni-osnabrueck.de/~koks/cgi-bin/hitle.rb";
36     private String baseURL;
37     private static final int MAX = 10;
38
39     boolean isStandalone = false;
40     TextArea textArea1 = new TextArea(5, 100);
41     TextArea textArea2 = new TextArea(2, 100);
42     TextField textArea3 = new TextField(Integer.toString(MAX), 2);
```

```

43     Button button1 = new Button();
44     Button button2 = new Button();
45     BorderLayout borderLayout1 = new BorderLayout();
46 //Variable zum Sprachenumschalten (dtl./engl.)
47     Choice cSprache = new Choice();
48 //protokollieren ja/nein
49     Checkbox checkBox = new Checkbox("pro", false);
50     /**
51      * getParameter
52      * @param key
53      * @param def
54      * @return java.lang.String
55      */
56     public String getParameter(String key, String def) {
57         if (isStandalone) {
58             return System.getProperty(key, def);
59         }
60         if (getParameter(key) != null) {
61             return getParameter(key);
62         }
63         return def;
64     }
65
66     /**
67      * Constructs a new instance.
68      */
69     public KoksAppe() { }
70
71     /**
72      * init
73      */
74     public void init() {
75         try {
76             jbInit();
77         }
78         catch (Exception e) {
79             e.printStackTrace();
80         }
81     }
82
83     /**
84      * Initializes the state of this instance.
85      */
86     private void jbInit() throws Exception {
87         // URL fuer Server
88 //URL zum 1. Ruby-Script
89         baseURL = getParameter("baseURL",
90 "http://nulix.cl-ki.uni-osnabrueck.de/~koks/cgi-bin/hitle.rb");
91         // Layout und Farbe
92         this.setLayout(borderLayout1);
93         this.setBackground(Color.lightGray);
94         // set Font
95         int size = Integer.parseInt(getParameter("fontsize", "24"));
96         Font font = new Font("SansSerif", Font.PLAIN, size);
97         this.setFont(font);
98         // Start Button
99         button1.setLabel("Start");
100        //button1.setBackground(Color.green);
101        button1.addActionListener(new java.awt.event.ActionListener() {
102            public void actionPerformed(ActionEvent e) {
103                button1_actionPerformed(e);

```



```

104     }
105   });
106   // Reset Button
107   button2.setLabel("Reset");
108   //button2.setBackground(Color.red);
109   button2.addActionListener(new java.awt.event.ActionListener() {
110     public void actionPerformed(ActionEvent e) {
111       button2_actionPerformed(e);
112     }
113   });
114   // Eingabefeld für Sätze
115   //textAreal.setText("Wir werden jemanden in Kenntnis setzen.");
116   textAreal.setBackground(Color.white);
117   // Ausgabefeld für markierten Satz
118   //textArea2.setBackground(Color.white);
119   //textArea2.setEditable(false);
120   // DropDown für Sprachauswahl
121   cSprache.add("G");
122   cSprache.add("E");
123   // unteres Panel enthält Button und DropDown
124   Panel unten = new Panel(new GridLayout(1, 2));
125   Panel links = new Panel(new GridLayout(1, 6, 8, 8));
126   links.add(new Label(VERSION));
127   links.add(new Label("lang: "));
128   links.add(cSprache);
129   links.add(new Label("hits: "));
130   links.add(textArea3);
131   links.add(checkBox);
132   Panel rechts = new Panel(new GridLayout(1, 2));
133   rechts.add(button2);
134   rechts.add(button1);
135   unten.add(links);
136   unten.add(rechts);
137   // alles einbauen
138   this.add(unten, BorderLayout.SOUTH);
139   this.add(textAreal, BorderLayout.CENTER);
140   //this.add(textArea2, BorderLayout.CENTER);
141 }
142
143 /**
144  * getAppletInfo
145  * @return java.lang.String
146  */
147 public String getAppletInfo() {
148   return "Applet Information";
149 }
150
151 /**
152  * getParameterInfo
153  * @return java.lang.String[][]
154  */
155 public String[][] getParameterInfo() {
156   return null;
157 }
158
159 void button2_actionPerformed(ActionEvent e) {
160   //textAreal.setText("Wir werden jemanden in Kenntnis setzen");
161   textAreal.setText("");
162   //textArea2.setText("");
163   textArea3.setText(Integer.toString(MAX));
164 }

```

```

165
166 void button1_actionPerformed(ActionEvent e) {
167     // Satz herausfiltern zwischen .!?"
168     int anfang, ende, max = MAX;
169     String text = textAreal.getText();
170     String selected = textAreal.getSelectedText();
171     String wort = new String(selected.trim());
172     String phrase, sprache;
173     int pos = textAreal.getCaretPosition();
174     int wortAnfang = textAreal.getSelectionStart();
175     int wortEnde = textAreal.getSelectionEnd();
176     String vorCursor = new String(text.substring(0, pos));
177     String nachCursor = new String(text.substring(pos, text.length()));
178
179     // Anfang und Ende vom Satz suchen
180     anfang = vorCursor.lastIndexOf(".") + 1;
181
182     if (anfang < vorCursor.lastIndexOf("!"))
183         anfang = vorCursor.lastIndexOf("!") + 1;
184     if (anfang < vorCursor.lastIndexOf("?"))
185         anfang = vorCursor.lastIndexOf("?") + 1;
186     if (anfang < vorCursor.lastIndexOf("'"))
187         anfang = vorCursor.lastIndexOf("'") + 1;
188
189     ende = vorCursor.length() + nachCursor.indexOf(".") + 1;
190
191     if ((ende > nachCursor.indexOf("!")) &&
192         (nachCursor.indexOf("!") != -1))
193         ende = nachCursor.indexOf("!");
194     if ((ende > nachCursor.indexOf("?")) &&
195         (nachCursor.indexOf("?") != -1))
196         ende = nachCursor.indexOf("?");
197     if ((ende > nachCursor.indexOf("'")) &&
198         (nachCursor.indexOf("'") != -1))
199         ende = nachCursor.indexOf("'");
200
201     //int wortAnfang = textAreal.getSelectionStart() - anfang;
202     //int wortEnde = textAreal.getSelectionEnd() - ende;
203     //Phrase extrahieren
204     phrase = cleanString(new String(text.substring(anfang, wortAnfang) +
205         "<i> " + text.substring(wortAnfang, wortEnde) + " </i> " +
206         text.substring(wortEnde, ende)));
207
208     //textArea2.setText("Anfang: " + anfang + " Ende: " + ende + "\n" +
209     // "WortAnfang: " + wortAnfang + " WortEnde: " + wortEnde + "\n" +
210     // "SelStart: " + textAreal.getSelectionStart() + " SelEnd: " +
211     // textAreal.getSelectionEnd() + "\n");
212     //Satz markieren
213     //textAreal.select(anfang, vorCursor.length() + ende);
214     textAreal.select(anfang, ende);
215     //textArea2.setText("Wort: " + wort + "\nPhrase: " + phrase);
216
217     // Auswahl von Sprache auswerten
218     sprache = new String(cSprache.getSelectedItem() == "G" ? "de" : "en");
219     if (!textArea3.getText().equals(""))
220
221     // max. Trefferanzahl auswerten
222     try {
223         max = Integer.parseInt(textArea3.getText());
224     } catch (NumberFormatException nfe) {
225         max = MAX;

```

```

226     }
227     max = (max <= 0) ? MAX : max;
228
229     // URL öffnen und in neuem Frame anzeigen
230     try {
231         getAppletContext().showDocument(createURL(phrase, sprache, max),
232 "ausgabe1");
233     }
234     catch (Exception ex) {
235         //textArea2.setText("Exception: " + e.toString());
236         ex.printStackTrace();
237     }
238 }
239
240 private URL createURL(String phrase, String sprache, int max) throws Exception
241 {
242 //erstellt eine neue URL mit den Parametern
243     String verbose = checkBox.getState() ? "1" : "0";
244     URL url = new URL(baseURL + "?port=29295&phrases="+
245 URLEncoder.encode(phrase) + "&max=" + max +
246 "&t=t:segment&s=s:text&applet=true&lang=" + sprache + "&verbose=" + verbose);
247     return url;
248 }
249
250 private String cleanString(String phrase) {
251     String result = phrase.replace('\t', ' ').replace('\n', ' ').trim();
252     return result;
253 }
254
255 }

```

P.1.2 hit1e.rb

Anfrage vom Applet bearbeiten (s. 10.3)

```

1 #!/usr/local/bin/ruby
2
3 # $Header: /opt/cvs/doc/phase_1/demo/hit1e.rb,v 1.2 2001/10/23 17:27:57 koks-cvs Exp $
4
5 # hit1e.rb - Process queries from web frontend and ask
6 #     demo-qdb for results
7 #     Adapted version for HIT
8 #     Adapted version for EuroCALL
9 #
10 # KOKS project
11 # Arno Erpenbeck, 06.08.2001
12 # english by Norman Kummer, 21.08.2001
13
14 require "cgi"
15 require "socket"
16 require "base64"
17 require "md5"
18 require "secret"
19
20 $dbURL = "nulix.cl-ki.uni-osnabrueck.de"
21 $cgiURL = "http://nulix.cl-ki.uni-osnabrueck.de/~koks/cgi-bin/hit2e.rb"
22
23 class Ergebnis
24     include Comparable
25
26     attr_reader :nr, :id, :type, :lemmata

```

```

27 attr_accessor :bid
28
29 def initialize(nr, id, type="unknown")
30   @nr = nr
31   @id = id
32   @type = type
33   @tokens = []
34   @lemmata = []
35 end
36
37 def size
38   @lemmata.size
39 end
40
41 def <=>(anOther)
42   self.size <=> anOther.size
43 end
44
45 def add_lemma(lem)
46   @lemmata << lem
47 end
48
49 def add_token(tok)
50   @tokens << tok
51 end
52
53 def to_s
54   str = "<tr><td valign=\"top\">\n"
55   if @type == "wb"
56     str << "<input type=\"hidden\" name=\"lemmata_#{@nr}\" value=\"\" +
57 @lemmata.join(" ") + "\">\n"
58   elsif @type == "erkannt"
59     str << "<input type=\"hidden\" name=\"bid_#{@nr}\" value=\"#{@id}\">\n"
60   end
61   str << "<input type=\"hidden\" name=\"type_#{@nr}\" value=\"#{@type}\">\n"
62   str << "<input type=\"radio\" name=\"select\" value=\"#{@nr}\"></td>\n"
63   str << "<td><tt>"
64   @tokens.each do |tok|
65     str << "#{CGI.escapeHTML(tok)} "
66   end
67   str << "</tt></td></tr>\n"
68   str
69 end
70
71 end
72
73 def askServer(port, cmd=nul)
74   begin
75     # open socket to specified server
76     socket = TCPSocket.open($dbURL, port)
77     # wait for challenge
78     question = socket.gets
79     what = question.split("'")[1]
80     # compute secret and send to server
81     socket.print what.computeSecret, "\n"
82     socket.flush
83     resp = socket.gets
84     if resp.strip! != "OK"
85       raise "Authentication not accepted"
86     end
87     # send command

```

```

88  socket.print cmd
89  socket.flush
90  puts "<form name=\"myform\" action=\"#{CGIURL}\" target=\"ausgabe2\"
91  method=\"post\">"
92  puts "<input type=\"hidden\" name=\"port\" value=\"#{port}\">"
93  puts "<input type=\"hidden\" name=\"t\" value=\"t:segment\">"
94  puts "<input type=\"hidden\" name=\"s\" value=\"s:text\">"
95  # read until finished
96  count = 1
97  id = ""
98  type = ""
99  modus = :anfang
100 lemma = []
101 erkannt = []
102 wb = []
103 ergebnis = nil
104 while resp = socket.gets.strip
105   if resp == "<DONE>" || resp == "<ERROR>"
106     break
107   elsif resp == "<SATZ>" || resp == "<segmentgrenze>" ||
108     resp == "<unbestimmter Regionentyp>"
109     if ergebnis.type == "erkannt"
110       erkannt << ergebnis
111     elsif ergebnis.type == "wb"
112       wb << ergebnis
113     else
114       end
115     count += 1
116     modus = :warte
117     lemma = []
118   elsif resp =~ /(<id:>(\d+) (type:)(\w+)/
119     id = $2
120     type = $4
121     ergebnis = Ergebnis.new(count, id, type)
122   elsif resp =~ /(<id:>(\d+)/
123     id = $2
124     ergebnis = Ergebnis.new(count, id, type)
125   else
126     tokens = resp.split('\t')
127     ergebnis.add_token(tokens[0])
128     ergebnis.add_lemma(tokens[2])
129   end
130 end
131
132 puts "<h4>according to examples</h4>"
133 puts "<table border=\"0\">"
134 erkannt.sort.each do |e| puts e end
135 puts "</table>"
136
137 puts "<h4>according to the dictionary</h4>"
138 puts "<table border=\"0\">"
139 wb.sort.each do |w| puts w end
140 puts "</table>"
141
142 puts "<br>number of hits: <input type=\"text\" size=\"10\" name=\"max\"
143 value=\"10\">"
144 puts "&nbsp;<input type=\"submit\" value=\"look at it\">\n</form>"
145 # signal end of data
146 socket.print "quit\n"
147 socket.flush
148 rescue

```

```

149     puts "<font color=\red\>Error connecting to server: #{$!}</font>"
150     puts "</body></html>"
151     return
152   ensure
153     socket.close if socket
154   end
155 end
156
157 # print out some info
158 cgi = CGI.new
159 print "Content-type: text/html\r\n\r\n"
160 puts "<html><head><meta http-equiv=\expires\ content=\0\></head>"
161 puts "<body bgcolor=\white\>"
162
163 # process form values
164 verbose = cgi.params["verbose"].to_s == "1"
165 port = cgi.params["port"].to_s
166 if cgi.params["wort"].to_s.length == 0 && cgi.params["phrases"].to_s.length == 0
167   puts "<font color=\red\>Error: look up for the empty word!</font>"
168   puts "</body></html>"
169   exit
170 end
171
172 cmd = ""
173 cmd << cgi.params["t"].to_s + ";"
174 cmd << cgi.params["s"].to_s + ";w:"
175 ["lang", "phrases", "id", "lemmata", "match", "name", "wort", "max",
176 "satzid"].each { |c|
177   if cgi.params[c].to_s.length > 0
178     str = cgi.params[c].to_s
179     str = str.delete(",") if c == "phrases"
180     while str.gsub!(/ /, " ")
181     end
182     cmd << c + "=" + str + ","
183   end
184 }
185 cmd.chomp! if cmd[-1,1] == ","
186
187 if verbose
188   puts "Port: <tt>", port, "</tt><br>"
189   puts "Command: <tt>", cmd, "</tt><br>"
190 end
191 puts "<h3>phrases found:</h3>"
192 STDOUT.flush
193
194 askServer(port.to_i, cmd)
195
196 # close document
197 puts "</body></html>"
198
199

```

P.1.3 hit2e.rb

Anfrage vom Applet bearbeiten (s. 10.3)

```

1 #!/usr/local/bin/ruby
2
3 # $Header: /opt/cvs/align/frontend/hit2.rb,v 1.1 2001/06/21 13:28:48 koks-cvs
4 Exp $
5

```

```

6 # hit2.rb - Process queries from web frontend and ask
7 #         demo-qdb for results
8 #         Adapted version for HIT
9 #         Adapted version for EuroCALL
10 #
11 # KOKS project
12 # Arno Erpenbeck, 06.08.2001
13 # english by Norman Kummer, 21.08.2001
14
15 require 'cgi'
16 require 'socket'
17 require 'base64'
18 require 'md5'
19 require 'secret'
20
21 $dbURL = 'nunix.cl-ki.uni-osnabrueck.de'
22
23 def askServer(port, cmd=nil)
24   begin
25     # open socket to specified server
26     socket = TCPSocket.open($dbURL, port)
27     # wait for challenge
28     question = socket.gets
29     what = question.split('')[1]
30     # compute secret and send to server
31     socket.print what.computeSecret, '\n'
32     socket.flush
33     resp = socket.gets
34     if resp.strip! != 'OK'
35       raise 'Authentication not accepted'
36     end
37     # send command
38     socket.print cmd
39     socket.flush
40     # read until finished
41     count = 1
42     id = ''
43     lemma = []
44     deutsch = true
45     while resp = socket.gets.strip
46       if resp == '<DONE>' || resp == '<ERROR>'
47         break
48       elsif resp == '<SATZ>' || resp == '<segmentgrenze>' ||
49         resp == '<unbestimmter Regionentyp>'
50         count += 1
51         lemma = []
52       elsif resp =~ /(<id:>(\d+)>)/
53         id = $2
54         puts deutsch ? '<br><br>german: ' : '<br>english: '
55         deutsch = !deutsch
56       else
57         tokens = resp.split('\t')
58         lemma << tokens[2]
59         puts '<tt>#{CGI.escapeHTML(tokens[0])} </tt>'
60       end
61     end
62     # signal end of data
63     socket.print 'quit\n'
64     socket.flush
65   rescue
66     puts '<font color=\<span style="color:red">Error connecting to server: #{$!}</font>'

```

```

67     puts '</body></html>'
68     return
69     ensure
70     socket.close if socket
71     end
72 end
73
74 # print out some info
75 cgi = CGI.new
76 print 'Content-type: text/html\r\n\r\n'
77 puts '<html><head><meta http-equiv='\"'expires\"' content='\"'0\"'></head>'
78 puts '<body bgcolor='\"'white\"'>'
79
80 # process form values
81 port = cgi.params['port'].to_s
82 select = cgi.params['select'].to_s.to_i
83 bid = cgi.params['bid_#{select}'].to_s
84 lemmata = cgi.params['lemmata_#{select}'].to_s
85 if bid.length == 0 && lemmata.length == 0
86     puts '<font color='\"'red\"'>Error: look up without BID or lexical item!</font>'
87     puts '</body></html>'
88     exit
89 end
90
91 cmd = ''
92 cmd << cgi.params['t'].to_s + ';'
93 cmd << cgi.params['s'].to_s + ';w:'
94 ['lang', 'phrases', 'id', 'lemmata', 'match', 'name', 'wort', 'max',
95 'satzid'].each { |c|
96     if cgi.params[c].to_s.length > 0
97         str = cgi.params[c].to_s
98         while str.gsub!(/ /, ' ')
99             end
100        cmd << c + '=' + str + ', '
101    end
102 }
103
104 if select > 0
105     if lemmata.length > 0
106         #cmd << 'match_l=' + lemmata
107         cmd << 'lemmata=' + lemmata
108     else
109         cmd << 'bid=' + bid if select > 0
110     end
111 end
112 cmd.chomp! if cmd[-1,1] == ','
113
114 #puts 'Command: <tt>' + cmd, '</tt><br>'
115 puts '<h3>translations found:</h3>'
116 STDOUT.flush
117
118 askServer(port.to_i, cmd)
119
120 # close document
121 puts '</body></html>'
122
123
124

```