Klassifizierung von Dokumenten mit Clustering- und Machinelearning-Verfahren

Jan-Ole Esleben
esleben@cl.uni-heidelberg.de

Titus v. d. Malsburg
malsburg@cl.uni-heidelberg.de

10. Februar 2002

Hauptseminar Information Retrieval im WS 01/02

Motivation

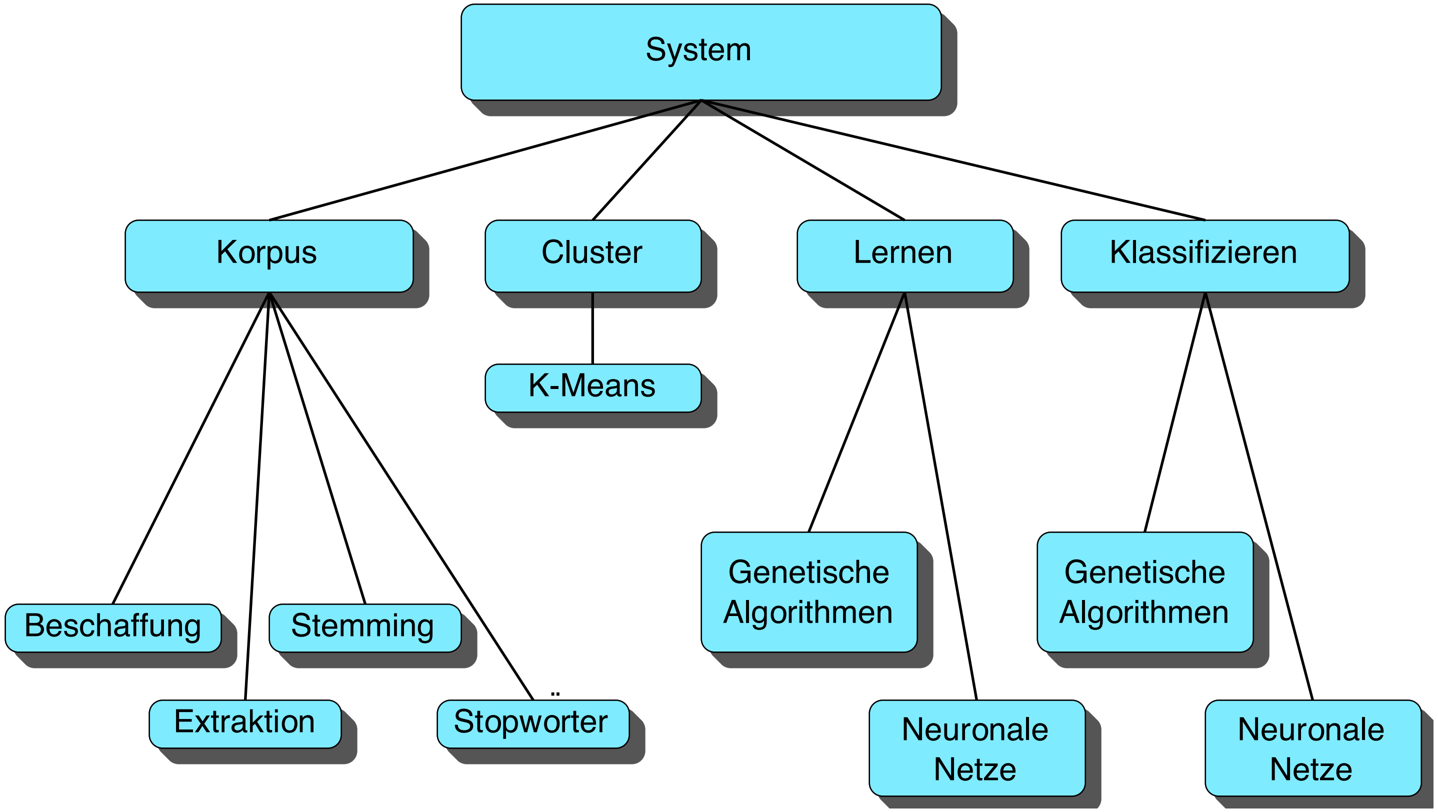
- Indexterme werden neben anderen Merkmalen zur Kassifikation von Dokumenten benutzt.
- Anzahl der Indexterme meistens gross.
- Lernalgorithmen (hier: Neuronale Netze, Genetische Algorithmen) skalieren schlecht.
- Clustering von Indextermen verringert die Anzahl der Attribute, die die Lerner verarbeiten müssen.

Kann Clustering der Indexterme Effizienz der Lerner steigern?

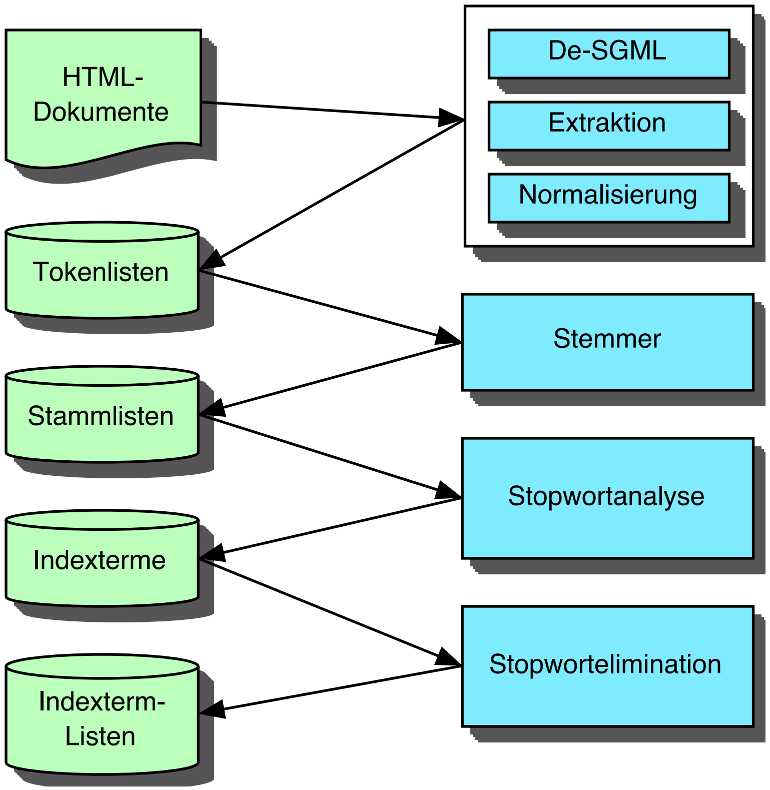
Übersicht

- Korpus: Online Ausaben von ”Die Zeit” des Jahres 2001 als HTML-Seiten
- Implementierung: Plattformunabhängig, in C, Python, Ruby, Java
- Kein monolithisches Werkzeug, sondern kleine Programme
- Datenaustausch über einfache Textdateien

Komponenten



Korpus

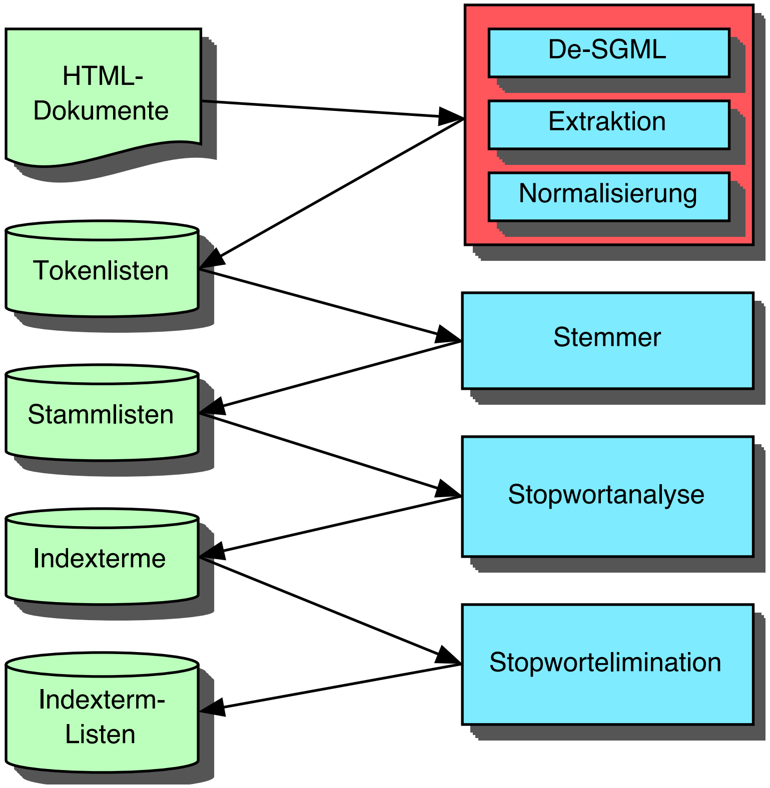

HTML-
Dokumente
De-SGML
Extraktion
Normalisierung
Tokenlisten
Stemmer
Stammlisten
Stopwortanalyse
Indexterme
Stopwortelimination
Indexterm-
Listen

Orginaldaten

```
<img src="../../../bilder/a.gif" width="1" height="12"
alt="" border="0"><br>
<div class="margin">
<font face="Verdana,Geneva,Arial,Helvetica,
sans-serif" size="2">James Harris, General Manager des
Aladdin Hotel-Resorts, greift zum Handy: <I>Hi, </I>
Howard, du kannst uns jetzt das Gewitter schicken. Ich
bin mit meiner Gruppe am Hafen. Und schon geht’s los:
Der Himmel verdunke...
```

Jan-Ole Esleben, Titus v. d. Malsburg

Aufbereitung



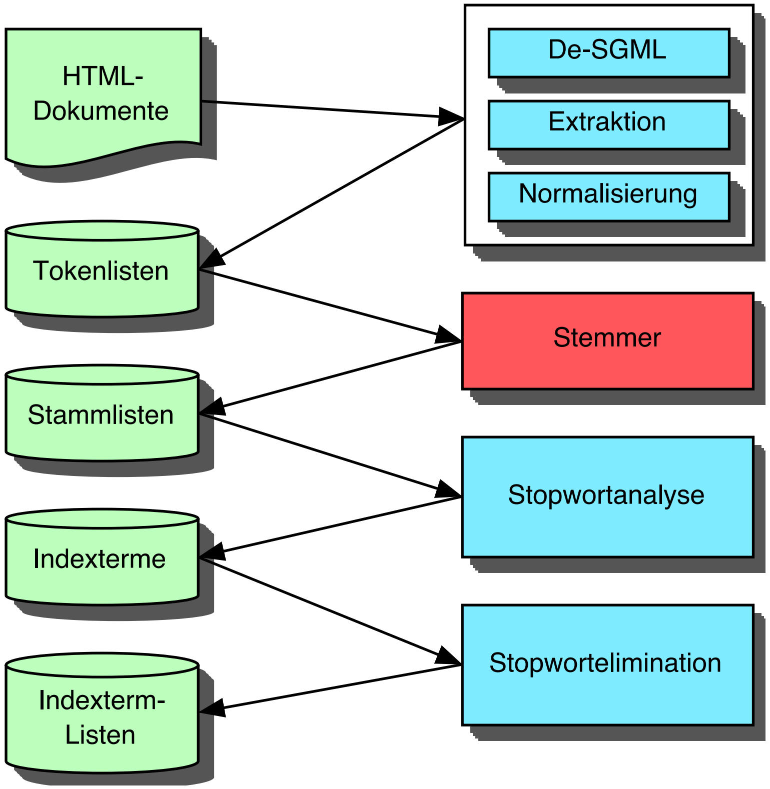
Aufbereitung

- De-SGML extrahiert Markup und ersetzt Entities
- Extraktion entfernt Navigation, Werbung, etc
- Normalisierung erzeugt Wortlisten
- Alle drei sind Lex-Programme

Tokenlisten

James	du	Und
Harris	kannst	schon
General	uns	gehts
Manager	jetzt	los
des	das	Der
Aladdin	Gewitter	Himmel
Hotel-Resorts	schicken	verdunkelt
greift	Ich	sich
zum	bin	Blitze
Handy	mit	zucken
Hi	meiner	
Howard	Hafen	

Stemming


HTML-
Dokumente
De-SGML
Extraktion
Normalisierung
Tokenlisten
Stemmer
Stammlisten
Stopwortanalyse
Indexterme
Stopwortelimination
Indexterm-
Listen

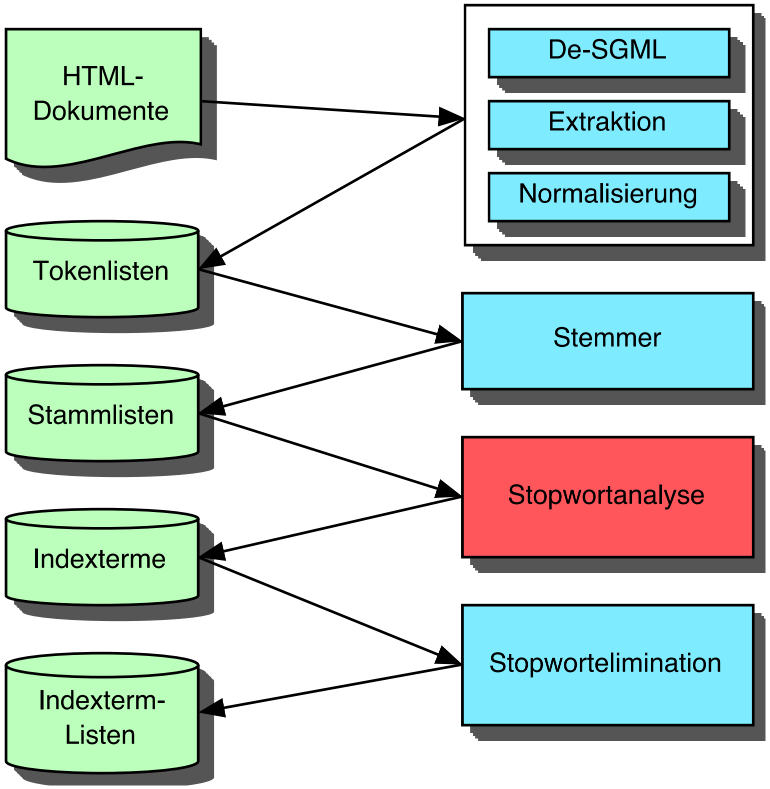
Jan-Ole Esleben, Titus v. d. Malsburg

Stämme

James	kann	Und
Harris	uns	schon
General	jetzt	gehts
Manag	das	los
des	Gewitt	D
Aladdin	schick	Himmel
Hotel-Resort	Ich	verdunkelt
greift	bin	sich
zum	mit	Blitz
Handy	mein	zuck
Hi	Grupp	Donnergroll
Howard	am	Dann
du	Haf	

Jan-Ole Esleben, Titus v. d. Malsburg

Identifikation von Stopwrtern mit Hilfe der Informationstheorie



Identifikation von Stopwrtern mit Hilfe der Informationstheorie

- Der Korpus beinhaltet ca. 230.000 verschiedene Stämme.
- Die Anzahl der Indexterme sollte um Grössenordnungen kleiner sein, da die Lerner sonst praktisch nicht arbeiten.
- Viele Tokens haben für unseren Zweck keine Bedeutung.

Idee 1

In jedem Dokument wird der Autor angegeben, sodass in jedem Dokument ”von” vorkommt.

Annahme: ”von” kann nicht als ein Indiz für die Zugehörigkeit zu einer best. Dokumentenklasse dienen.

Wörter wie ”die”, ”und”, ”ein”, ”nicht”, ”an”, ... kommen sehr oft vor und haben für uns keinen Nutzen.

Annahme: Tokens, die in nur fast allen Dokumenten vorkommen, sind ebenfalls nicht geeignet.

Idee 2

- Wörter die nur einmal im Korpus vorkommen, sind gute Indikatoren, sie können nur leider fast nie verwendet werden.

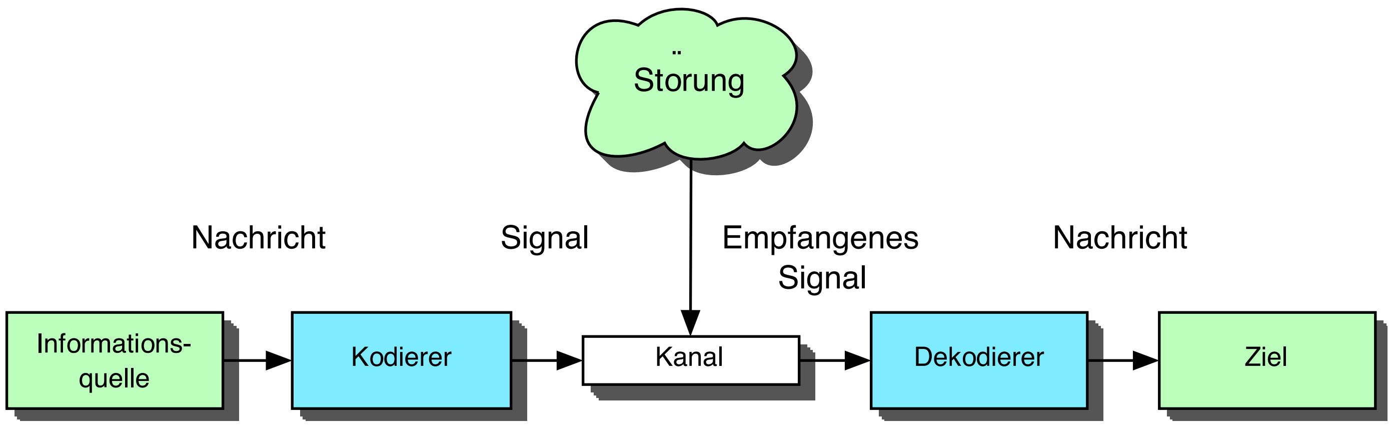
- Der Korpus enthält über 100.000 Wörter, die nur einmal vorkommen!

- Beispiele: ”abba-Nachahmungstrauma”, ”abc-Abwehrfahig”, ”Abnickverein”, ”Raffgierkapitalismus”, ”Ökoromantik”, ”Zwitterwelt”, ”Zwiebelturmvilla”, ...

$\Rightarrow$ Aus Effizienzgründen will man diese Tokens alle loswerden.

Jan-Ole Esleben, Titus v. d. Malsburg

Shannon’sche Informationstheorie: Model



Information: Dokumentenklasse
Kodiert: Als ein Dokument

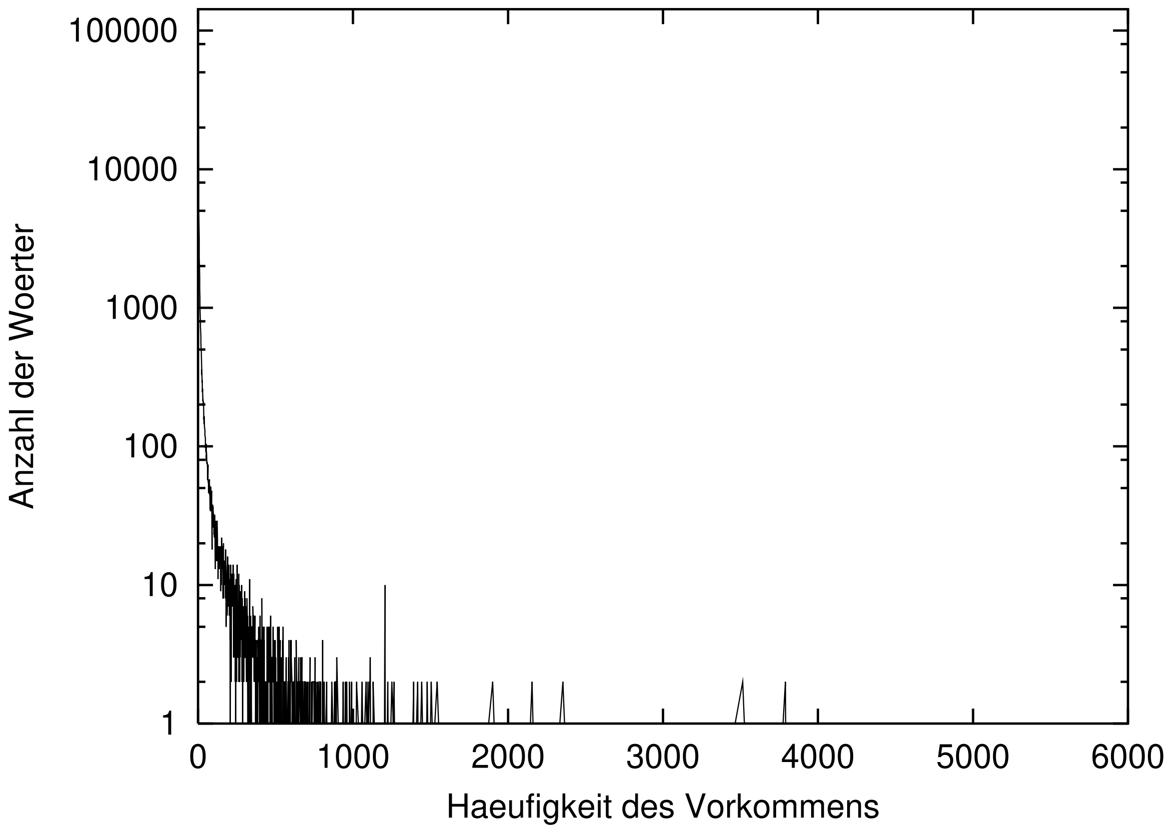
Informationsgehalt

- Haben alle Kodewörter ein gemeinsames Teilwort, so ist der Kode redundant.
- Der Informationsgehalt dieses Teilworts ist dann 0.
- Der Informationsgehalt ist gegeben durch

$$H(p_1, ..., p_n) = \sum_{i=1}^{n} p_i \log p_i$$

Jan-Ole Esleben, Titus v. d. Malsburg

Schwellenwerte



Algorithmus

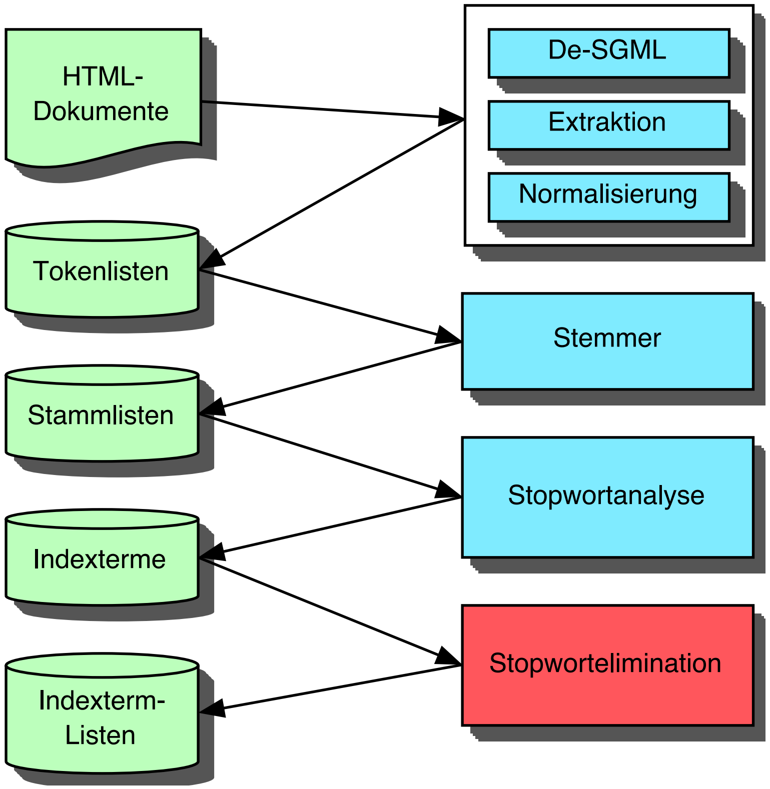
```
dict = {}
for d in documents:
  dhash = txt2dict(d)
  for w in dhash:
    dict[w] = dict[w] + 1

def txt2dict(d):
  dhash = {}
  for word in d:
    dhash[w] = 1
```

Eigenschaften

- Speicherverbrauch hauptsächlich von der Anzahl der versch. Wörter abhängig. Bei uns ca. 20 MB
- Komplexitätsklasse: $O(n)$
- Bei unserem Korpus 1-2 min

Jan-Ole Esleben, Titus v. d. Malsburg

Elimination von Stopwörtern



Elimination von Stopwörtern

- Python-Programm liest Indexterme in Feld `iterms`

```
for d in documents:
  d1 = []
  for word in d:
    if word in iterms:
      d1.append(word)
  write(d1)
```

Indexterme

james
general
manag
greift
handy
du
schick
grupp
himmel
reg
aufs
aufs
beck

band
frisch
wind
kunstlich
erzeugt
berrasch
gel
james
lachelt
innerhalb
nacht
las
gewalt

anstreng
bisher
herkomm
begriff
streck
jahrzehnt
sechs
kilomet
entstand
40
5000
bett
kunstlich

Ergebnisse 1

	Initial	Wortlisten	Stämme	Indexterme
Tokens	12.972.260	5.241.446	5.217.589	989.161
Types		313.085	231.160	3.056
KB	160.456	45.532	41.532	20.784

Ergebnisse 2

- Alle Verfahren sind $O(n)$.
- Der gesamte Korpus kann in wenigen Minuten verarbeitet werden.