



# Information Retrieval

## 05 – Index-Konstruktion

Philipp Schaer, Technische Hochschule Köln, Cologne, Germany

Version: 2022-04-28

# Was haben wir heute vor...?

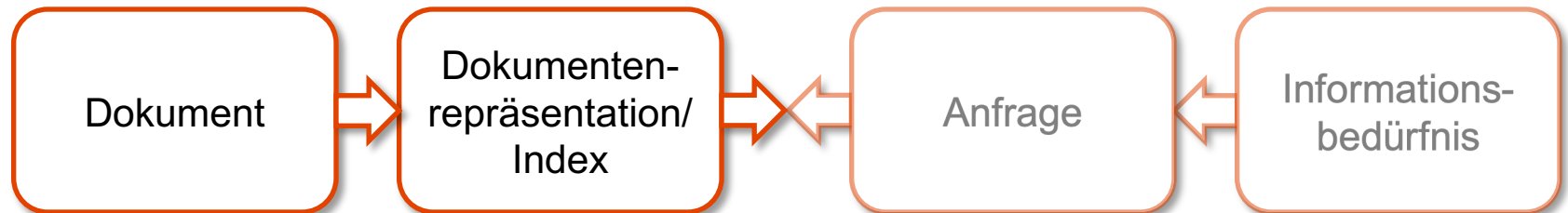
## Orga-Time

- Vektor-Muster-Lösung ist online.
- Erstes Quiz ist in Moodle online. Viel Erfolg.

## Leitfragen für heute

- Welche **zwei gegensätzlichen Ansätze** finden sich nahezu in allen Indexierungmodulen wieder? Gibt es hierbei **Pros** und **Cons**?
- Welche **linguistischen Module** gibt es, die den Anspruch an einen "best match" aktiv unterstützen?
- Und **wie genau machen** das die Module jeweils?
- Welche **Pipeline** auf Folie 38 gewinnt?

# Vorbemerkung zur Indexerstellung



**Grundfragen**, die vor der Indexerstellung geklärt sein sollten

- Welches **Vorverarbeitung** (Preprocessing) muss gegeben sein um die Term-Dokumentmatrix und damit das “**Vokabular**” aufzubauen?
- Welche **Terme** kommen in den Index?
- Einiges, was Sie hier hören kennen Sie ggf. auch schon von **Herrn Lepskys Veranstaltungen bzw. werden es dort kennenlernen!**

# Themen der Veranstaltung

- Indexierung – manuell vs. automatisch
- Grundlegende Indexierungspipeline
- Vorverarbeitung
- Indexerstellung – Invertierte Listen



Bur

Bunk, Gerhard P

Bunk, Gerhard P.:

Didaktik und Methodik des kaufmännisch-wirtschaftlichen  
Unterrichts in der neuen Fachliteratur.

In: Wirtschaft und Erziehung, 19(1967)1, S. 8-16.

Kaufmännische Berufserziehung.

DIz670193

p. 6(05)

Wir

Indexierung. Damals...

# Intellektuelle Indexierung (manuell)

Basis = Dokumentationssprache (Terminologiekontrolle)

- Intellektuelle **Interpretation** des Dokumentinhalts & Verschlagwortung
- Sehr **präzise** Einordnung & Recherche möglich
- Intellektuelle **Relevanzentscheidung** beim Indexieren & Recherchieren
- Ursprung **Bibliothekswesen**: notwendig bei systematischer, physischer Ablage

Nachteile

- Aufwendig (Entwicklung & Anwendung)
- Langsam, skaliert nicht (viele Dokumente → viel Aufwand)
- Lernaufwand für den Benutzer

# Automatische Indexierung

## Maschinelle Prozesse der Inhaltsanalyse und -darstellung

- Vollautomatisch
- Teilautomatisch
- Vorteil: Schnelligkeit
  
- Für **große Textmengen**: Zur Bewältigung der Informationsflut
- Es existieren aber typische **Problembereiche**:
  - Inhaltliche Interpretation?
  - Qualität ausreichend?
  - Nicht-textuelle Medien?

# Automatische Indexierung: Typologie

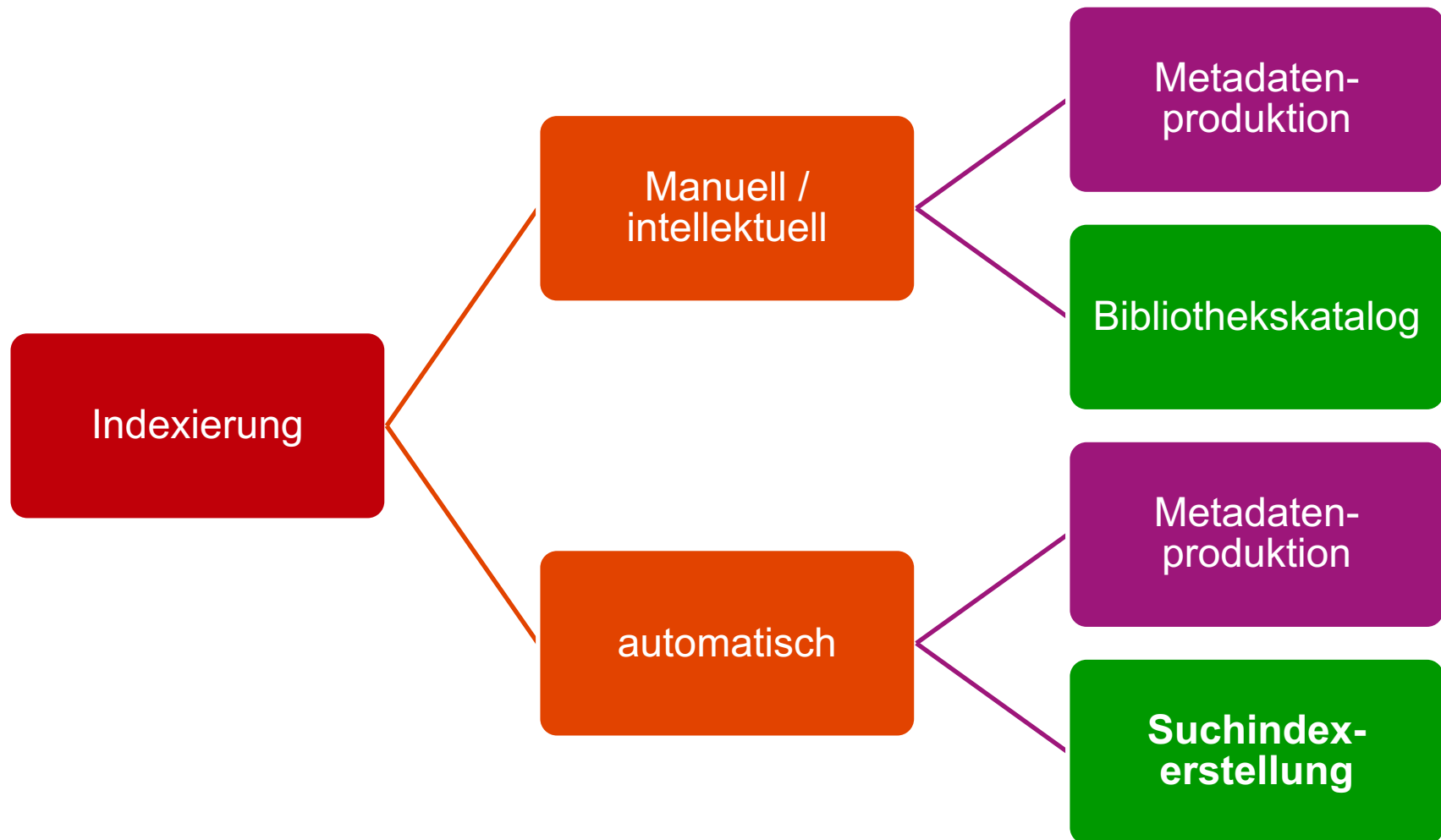
1. Verfahren, die **Metadaten** produzieren
  - Ausgehend von einer existierenden Dokumentationssprache
  - Ausgehend vom Dokument
2. Verfahren, die vorhandenen Text prozessieren und für die Suche bereitstellen

Typische Ansätze:

- **Computerlinguistisch:** Wortebene (Regeln, Wörterbücher)
- **Statistisch:** Worthäufigkeiten (weitverbreitete Methode)
- **Begriffsorientiert:** Begriffsebene



# Übersicht über Indexierungsverfahren



# Definitionen

## Wort

- Eine begrenzt lange Buchstabenreihung in einem Text.

## Term

- Ein normalisiertes Wort; eine Äquivalenzklasse von Wörtern.

## Token

- Eine Instanz / konkrete Nutzung eines Worts oder Terms in einem Dokument.

# Grundlegende Indexierungspipeline

Dokumente, die  
indexiert werden sollen



Friends, Romans, Countrymen.  
⋮

Tokenizer

Token-Stream

Friends

Romans

Countrymen

Linguistische Module

Modifizierte  
Tokens

friend

roman

countryman

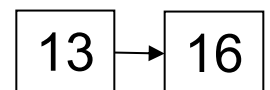
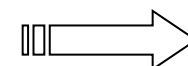
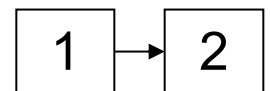
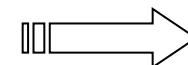
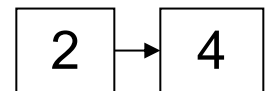
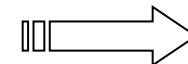
Indexer

Invertierte Liste / Index

**friend**

**roman**

**countryman**



# Indexer Verarbeitungsschritte

Erstellen einer Sequenz aus  
Term-Dokument-ID-Paaren.

Doc 1

I did enact Julius Caesar  
I was killed i' the Capitol;  
Brutus killed me.

Doc 2

So let it be with Caesar.  
The noble Brutus hath told  
you Caesar was ambitious



Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

# Indexer Verarbeitungsschritte

Term	Doc #
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2



Sortieren der  
Sequenz

Term	Doc #
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2



Mehrfache  
Begriffe  
werden  
**vereinigt**  
und die  
**Frequenz**  
gespeichert

Term	Doc #	Term freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1



# Aufteilung in Dictionary und Postings

Term	Doc #	Freq
ambitious	2	1
be	2	1
brutus	1	1
brutus	2	1
capitol	1	1
caesar	1	1
caesar	2	2
did	1	1
enact	1	1
hath	2	1
I	1	2
i'	1	1
it	2	1
julius	1	1
killed	1	2
let	2	1
me	1	1
noble	2	1
so	2	1
the	1	1
the	2	1
told	2	1
you	2	1
was	1	1
was	2	1
with	2	1

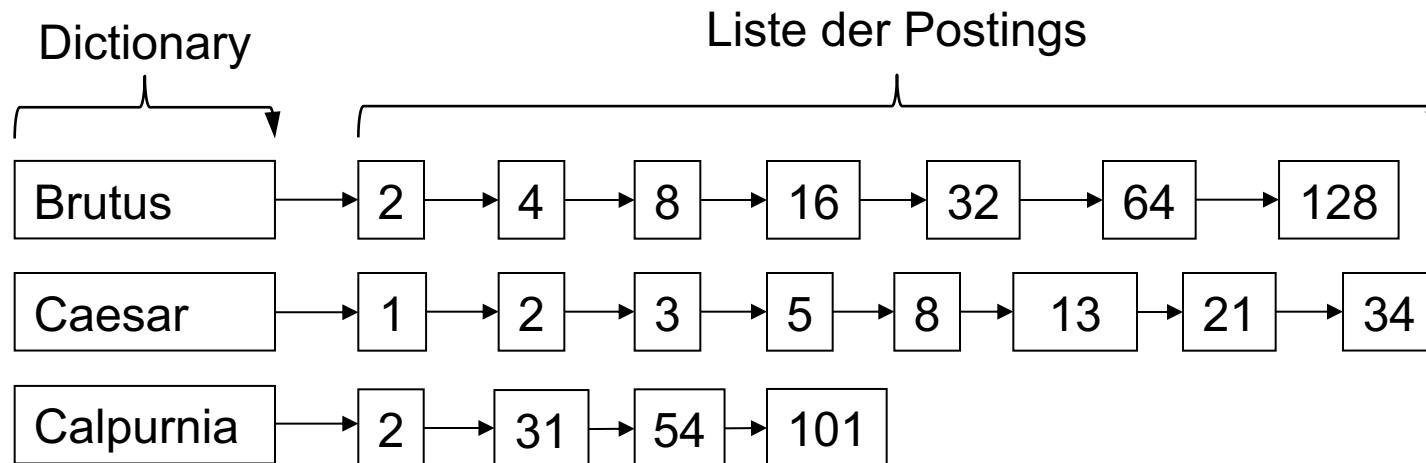


Term	N docs	Coll freq
ambitious	1	1
be	1	1
brutus	2	2
capitol	1	1
caesar	2	3
did	1	1
enact	1	1
hath	1	1
I	1	2
i'	1	1
it	1	1
julius	1	1
killed	1	2
let	1	1
me	1	1
noble	1	1
so	1	1
the	2	2
told	1	1
you	1	1
was	2	2
with	1	1

Doc #	Freq
2	1
2	1
1	1
2	1
1	1
1	1
2	2
1	1
1	1
2	1
1	2
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1
2	1
1	1
2	1
2	1

# Konstruktion invertierte Liste / Index

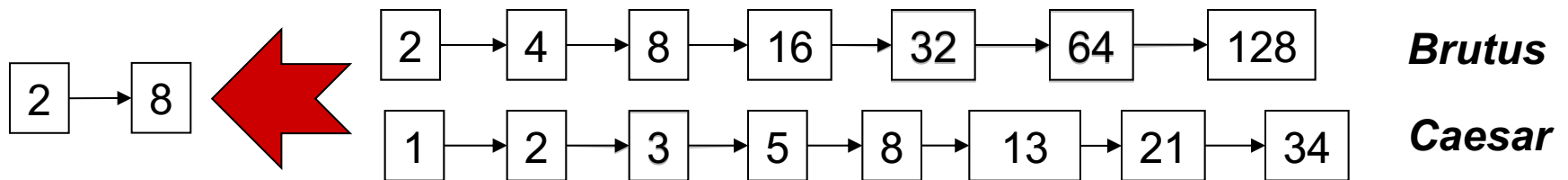
Für jeden Term  $t$  speichern wir die Dok-IDs, in denen  $t$  vorkommt.



**Übrigens!** Es ist entweder eine invertierte Liste **oder** ein Index; einen invertierten Index gibt es nicht (das wäre doppelt-gemoppelt).

# Eine Anfrage an den Index stellen

Wird eine Anfrage mit zwei Termen gestellt (Brutus AND Caesar), werden beide Postings simultan abgearbeitet und bei einer Übereinstimmung der Einträge das Ergebnis gespeichert.

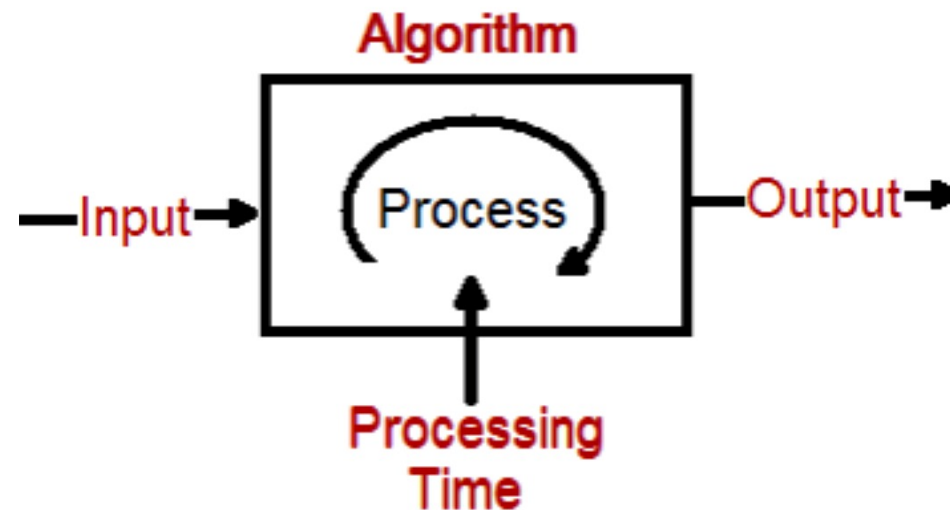


Wenn die Listen die Länge  $x$  und  $y$  haben, dann benötigt der Vergleich max.  **$O(x+y)$  Operationen.**

Wichtig: Hierzu müssen die Liste der Postings sortiert sein!

# Exkurs: O-Notation

- Die O-Notation beschreibt die Komplexität / Laufzeit eines Algorithmus
- Grundsätzlich muss man Eingabe (Input), Ausgabe (Output) und die Verarbeitungszeit (Processing Time) betrachten



# O-Notation: Beispiel

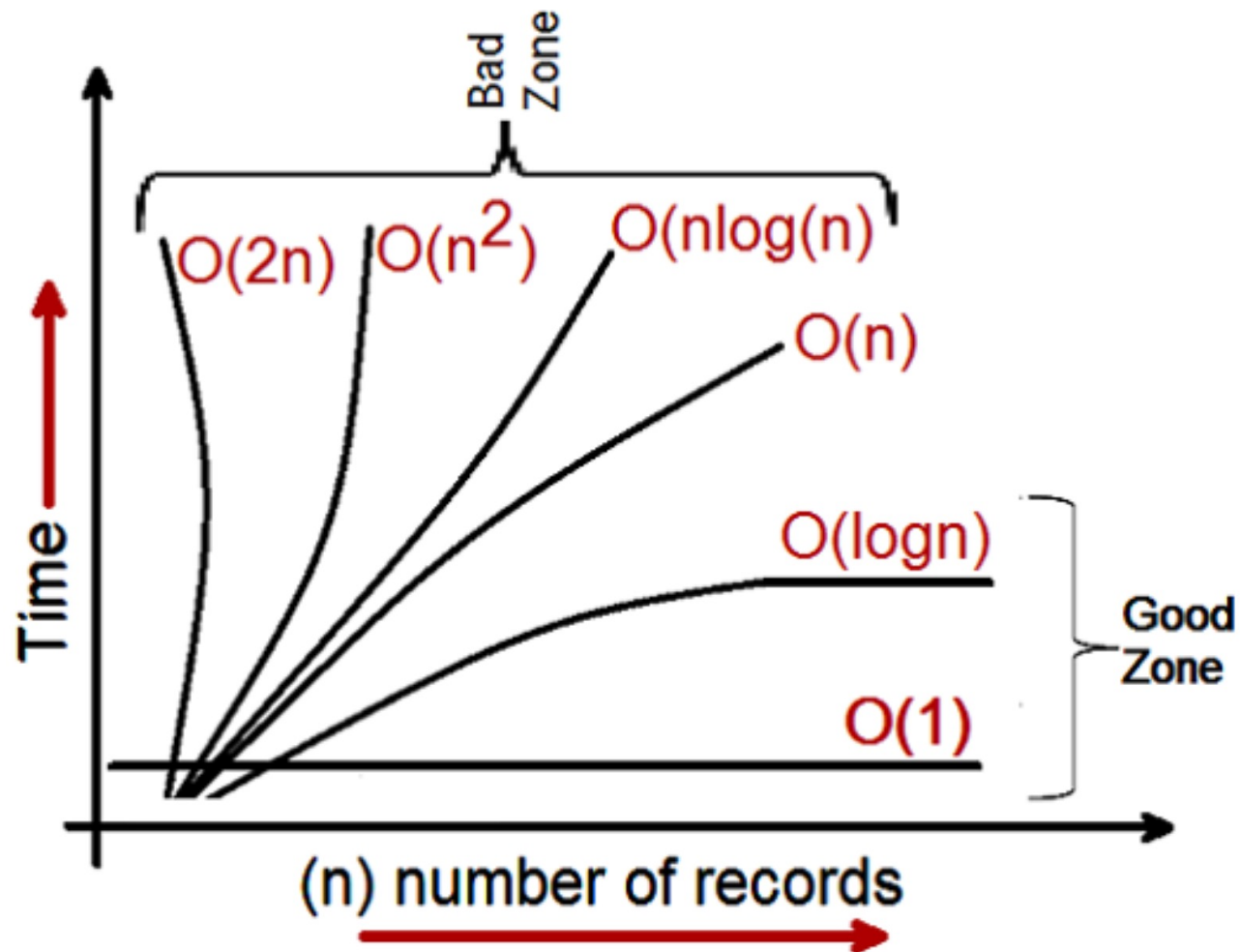
- Ein Algorithmus braucht zur Verarbeitung von Records (Datensätzen, Dokumenten, o.ä.) unterschiedlich lange.
- Aus einer Messung der Verarbeitungszeit unterschiedlich großer Mengen von Records ergibt sich eine Abschätzung von  $O(n^2)$ .
- *Ist das gut oder schlecht?*

5 Rec	→	27 Sec
10 Rec	→	105 Sec

$$\begin{aligned} &\Downarrow \\ \text{Time} &\cong (\text{Rec})^2 \\ &\Downarrow \\ &O(\text{Rec}^2) \\ &\Downarrow \\ &O(n^2) \end{aligned}$$



# O-Notation grafisch



# Automatische Indexierung: Probleme

- **Sprachliche Herausforderungen**
  - Paraphrasen
  - Umgangssprache
  - Metaphern
  - Anaphern (rhetorische Wortfigur: z.B. Wiederholung)
  - Neologismen
  - Synonyme
  - Polyseme/Homonymie
- **Gewichtung der Terme im Dokument**
- **Zuordnung zu einer Dokumentationssprache**

# Computerlinguistische Verfahren

Die Computerlinguistik bietet eine ganze Reihe an Verfahren:

- **Tokenisierung:** Segmentierung eines Textes auf Wortebene
- **Normalisierung:**
  - Satzzeichen-Entfernung: U.S.A. | USA
  - Groß-/Kleinschreibung
- Entfernung „nicht hilfreicher“ Terme → **Stoppworte**
- Vereinheitlichung unterschiedlicher Wortformen → **Stemming** / **Lemmatization**
- **Kompositazerlegung**
- **Phrasenerkennung**
- **Eigennamenerkennung**

# Stoppworte

... haben folgende Eigenschaften

- Inhaltslos
- Gleiche Wahrscheinlichkeit, in relevanten wie nichtrelevanten Dokumenten vorzukommen
- Werden aus dem Index bzw. der Anfrage entfernt (und folglich nicht weiter in Betracht gezogen)
- sprachabhängig

... kommen in unterschiedlichen Kontexten vor

- Allgemeine Stoppworte einer Sprache
- Stoppworte in einer gegebenen Dokumentmenge
- Stoppworte für einen gegebenen Zweck

# Stoppworte

## Erstellung

- Manuelle Liste
- Automatische Liste (Zipf)

## Typische Kandidaten:

- Artikel, Konjunktionen, Präpositionen, Hilfsverben

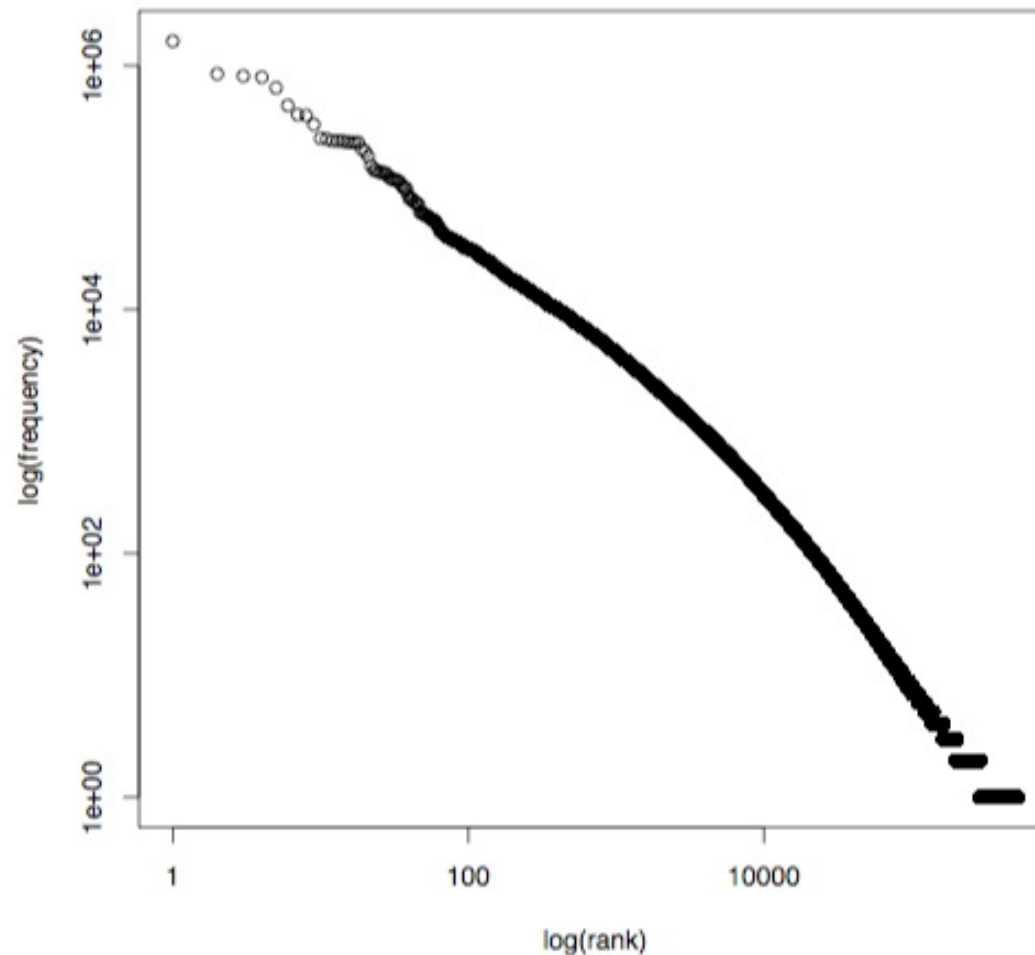
## Probleme:

„To be or not to be“

„I can open a can of tuna with a can opener.“



# Wo sind die Kandidaten für Stoppworte?



# Wortform-„Verschmelzung“ (conflation)

- Varianten eines Wortes auf eine **Grundform** zurückführen
- **Wortformen** tauchen damit in Gewichtungsberechnungen als **der gleiche Term** auf

Techniken:

- Nicht-linguistisch → **Stemming**
  - Retrieved, Retrieval → Retriev
- Linguistisch → **Lemmatization**
  - Retrieved, Retrieval → Retrieve

# Stemming (Wortstammbildung)

- Regeln, um Wortvarianten auf einen gemeinsamen Stamm zu reduzieren
- Sprachabhängig
- Meist Abtrennen von Suffixen
- (Präfixe = anti-, un- → Bedeutungsveränderung)

Verfahren:

- **Longest-match** → längste Endung wird in einem Schritt entfernt
- **Iterativ** → Aufeinanderfolgende, regelbasierte Entfernung / Umwandlung von Endungen
- **Korpus-basiert** → vermeidet Polysemproblem (Wortvarianten mit unterschiedlichen Bedeutungen kommen nicht im gleichen Dokument vor)

# Longest-Match Stemmer (Lovins, 1968)

**.11.**

alistically B  
arizability A  
izationally B

**.10.**

antialness A  
arisations A  
arizations A  
entialness A

**.09.**

allically C  
antaneous A  
antiality A  
arisation A  
arization A  
ationally B  
ativeness A  
eableness E  
entations A

**.09.—Cont.**

entiality A  
entialize A  
entiation A  
ionalness A  
istically A  
itousness A  
izability A  
izational A

**.08.**

ableness A  
arizable A  
entation A  
entially A  
eousness A  
ibleness A  
icalness A  
ionalism A  
ionality A

# Iterativ: Porter Stemmer (1980)

## Step 1a

SSES → SS

IES → I

SS → SS

S →

caresses → caress

ponies → poni

ties → ti

caress → caress

cats → cat

## Step 1b

(m > 0) EED → EE

(\*v\*) ED →

(\*v\*) ING →

feed → feed

agreed → agree

plastered → plaster

bled → bled

motoring → motor

sing → sing

## Step 1c

(\*v\*) Y → I

happy → happi

sky → sky

- m → Vokal-Konsonant Folge
- \*v\* → Stamm muss Vokal enthalten
- 5 Iterationen



# Overstemming / Understemming

## Understemming

- 2 Wortformen der gleichen Begriffsgruppe enden in unterschiedlichen Stems
- Divided division → divid divis

## Overstemming

- 2 Wortformen unterschiedlicher Begriffsgruppen enden in gleichen Stems
- Operate operating operates operation operative operatives operational → oper
- University universe → univers

# Lemmatisierung

Im Gegensatz zum automatischen Stemming wird bei der Lemmatisierung, die auf **morphologische Grundform** reduziert.

- Grundform: Nominativ Singular, Infinitiv Präsens etc.

## Verfahren

- **Regel-basiert** für Wortformen
- **Wörterbuch-basiert** (Erkennung Wortform + Wörterbuch nötig)

## Eigenschaften

- Vermeidet Over-/understemming
- Aufwendiger für die Implementation
- Jedoch: Stemming funktioniert in den meisten Fällen gut genug!

# Kompositazerlegung

## Beispiele

- Fußballweltmeisterschaft,
- Kompositazerlegung...

## Zerlegung in morphologische Bestandteile

- Erhöht Recall

## Verfahren:

- Wörterbuch-basiert
- Regelbasiert
  - Landschaft – s – malerei

# Phrasenerkennung

## Beispiele für Phrasen

- „White House“,
- „Deutsche Nationalbibliothek“,
- „Information Retrieval“

## Indexierung als ein Indexterm

- erhöht Precision

## Erweitert: Phrasenzusammenführung

- Information retrieval | retrieval of information

## Verfahren:

- Statistisch (Kookkurrenz)
- Syntaktisch (grammatikalische Struktur)
- Wörterbuch-basiert

# Soundex (Phonetische Korrektur)

## Alternatives Indexierverfahren, das auf Aussprache basiert

1. Behalte den ersten Buchstaben des Terms.
2. Entferne folgenden Buchstaben, bzw. ignoriere sie:  
A, E, I, O, U, H, W, Y.
3. Ändere folgende Buchstaben in Zahlen:
  1. B, F, P, V → 1
  2. C, G, J, K, Q, S, X, Z → 2
  3. D, T → 3
  4. L → 4
  5. M, N → 5
  6. R → 6
4. Löse immer wieder gleiche Zahlenpaare auf.
5. Fülle das Ergebnis mit Nullen auf, damit insgesamt 4 Zeichen entstehen (1 Buchstabe, gefolgt von 3 Zahlen)

# Soundex

## Beispiele

- Britney → BRTN → **B635**
- Spears → SPRS → **S162**
- Superzicke → SPRZCK → S16222 → **S162**
- Ähnlich-klingende Worte werden gleichgesetzt
- Gut für Namensvarianten (Müller, Mueller, Miller)

## Allerdings...

- ist Soundex sehr auf die englische Sprache ausgerichtet.  
(Klappt im Deutschen nicht sonderlich zuverlässig.)
- ist der Laut-Vergleich nur sehr grob.

# Eigennamenerkennung

Was passiert, wenn wir Eigennamen nicht erkennen?

- Heath Ledger (machinell übersetzt) → Heide Kantholzträger
  - Julia Roberts (Stemming) → Julia Robert
  - Old Shatterhand, New York → ?
- 
- Wir wollen die Indexierung **als ein Indexterm**
    - erhöht Precision
    - Erweitert: Namensdisambiguation

Verfahren:

- Normdateien
- regelbasiert

# Fehlertoleranz / Rechtschreibkorrektur



## Typische Probleme

- Falscher Buchstabe
- Ausgelassener Buchstabe
- Buchstabendreher
- Zusätzlicher Buchstabe
- „Alles andere“

## Lösungsansätze

- „nächste“ Alternative
- häufigste Alternative



# Levenshtein-Distanz (Editierabstand)

Anzahl von Einfügen, Löschen und Ersetzen Operationen, um eine Zeichenkette in eine andere zu überführen

- Tier
- Toer (Ersetze i durch o)
- Tor (Lösche e)
- → **Levenshtein-Distanz ist 2!**

Nächste Alternative =

- niedrigste Anzahl von Editieroperationen

Häufigste Alternative =

- welche Wortkombinationen kommen häufiger vor in der Dokumentensammlung

# Beispiel-Pipelines (wo steckt der Fehler)

**1. Tokenizer**

**2. Stemming**

**3. Normalisierung**  
(z.B. Groß-/Kleinschreibung,  
Sonderzeichen)

**4. Stoppwort-Entfernung**

**1. Tokenizer**

**2. Phrasenerkennung**

**3. Normalisierung**  
(z.B. Groß-/Kleinschreibung,  
Sonderzeichen)

**4. Stemming**

**5. Stoppwort-Entfernung**

# Zusammenfassung

- Ein Suchindex ist eine viel **effizientere Suchstruktur**, als eine Liste oder eine Term-Dokumentmatrix
- Die Indexierungspipeline arbeitet die nötigen Schritte ab
- Die Indexerstellung ist mit vielen **inhaltlichen** und **sprachlichen Problemen** behaftet für die es eine Reihe **intellektueller/ manueller** wie **technische/ statistische/ automatische Lösungen** gibt, z.B.
  - Stoppwortlisten,
  - Stemming / Lemmatization
  - Kompositazerlegung
  - Phrasenerkennung
  - Eigennamenerkennung
  - uvm.