

Korpora, Tokenisierung, Wortnormalisierungen, Wortverteilungen, Annotationen

Katja Markert

Institut für Computerlinguistik
Universität Heidelberg
`markert@cl.uni-heidelberg.de`
Einige Folien von Dan Jurafsky

November 14, 2022

- 1 Korpora
- 2 Tokenisierung
 - Wörter
 - Satzsegmentierung
- 3 Wortnormalisierungen
- 4 Wortverteilungen
- 5 Appendix (Optional): NLTK

- Zeichenketten
- Erkennung, Generierung, Stringvergleich
- Jetzt: Für empirische NLP und zum Lernen aus Texten braucht man Textsammlungen. Was für Textsammlungen gibt es?
- Jetzt: Wie teile ich einen Text überhaupt in bestimmte Zeichenketten wie Wörter und Sätze ein?
- Jetzt: Wie kann ich Wörter weiter normalisieren?
- Jetzt: Wie sehen Wortverteilungen in natürlichsprachlichen Texten aus?
- Jetzt: Welche weiteren Annotationen sind noch in Korpora enthalten?

- 1 Korpora
- 2 Tokenisierung
 - Wörter
 - Satzsegmentierung
- 3 Wortnormalisierungen
- 4 Wortverteilungen
- 5 Appendix (Optional): NLTK

Was ist ein Korpus?

das Korpus: (heutige) Definition

endliche Sammlung von maschinenlesbaren, natürlich vorkommenden Texten; oft nach bestimmten Kriterien selektiert, um ein bestimmtes Sprachproblem zu studieren

Man lernt verschiedene Dinge aus verschiedenen Texten!

Beispiele:

- Wortbedeutungen
- Wortsequenzen und Grammatik
- Textstruktur

- **Sprachtyp:** welche Sprache(n) von 7,151 lebenden Sprachen (www.ethnologue.com), Dialekte vs. “standard”
- **Genres:** Romane, Zeitungstexte, Dialog, ...
- **Domänen:** allgemein, Chemie, USA des 19. Jhdt, ...
- **Zeit** (synchron/diachron)
- **Autor/Sprecher**
- **Medium:** Textuell, Audio, Transkriptionen, Video.
- **Größe**
- Zusätzliche linguistische Annotationen? Tokenisierung, POS ...

And the Lord smelled a sweet savour, and said: I will no more curse the earth for the sake of man: for the imagination and thought of man's heart are prone to evil from his youth: therefore I will no more destroy every living soul as I have done. And the ark rested in the seventh month, on the seventh day of the...

Woher stammt dieser Textabschnitt?

"A certain selection and discretion must be used in producing a realistic effect," remarked Holmes. "This is wanting in the police report, where more stress is laid, perhaps, upon the platitudes of the magistrate than upon the details, which to an observer contain the vital essence of the whole matter. Depend upon it, there is nothing so unnatural as the commonplace."

Woher stammt dieser Textabschnitt?

Erste größere Korpora in den 60er Jahren: Brown and LOB

Brown Korpus

1 Million Wörter. Manuell mit Wortarten annotiert. Balanciertes Korpus des geschriebenen Amerikanischen Englisch.

LOB

Lancaster-Oslo/Bergen Korpus. Publizierte Texte. Britisches Englisch.

Balanciertes Korpus

Versucht, über Genres und Domänen repräsentativ zu sein.

Ist ein balanciertes Korpus eine gute Idee?

Korpus	Größe	Domäne	Sprache
Web	?	??	viele
Web 1T 5-gram (2006)	1 Billion	viele	Englisch
British National Corpus	100 Millionen	balanciert	Britisches Englisch
English Gigaword	1 756 504 000	newswire	Englisch
UN oder EU Proceedings	20 Millionen	rechtlich	10 Sprachpaare
Switchboard	2.4 Millionen	Dialog	US Englisch
DWDS Kernkorpus	100 Millionen	balanciert	Deutsch 20 Jhdt
Penn Treebank	1M	newswire	US Englisch

Links (die meisten auch auf ella unter resources z.b.

/resources/corpora/monolingual/raw/gigaword_eng_5):

- **Web 1T 5-gram:** <https://catalog.ldc.upenn.edu/LDC2006T13>
- **BNC:** <http://www.natcorp.ox.ac.uk/>. Besseres Interface unter <http://bncweb.lancs.ac.uk/bncwebSignup/user/login.php>
- **Brown corpus:** Teil von NLTK <http://www.nltk.org>
- **DWDS Korpora:** www.dwds.de

- Beispiel automatische Textzusammenfassung wie das CNN/DM Korpus:

Article: smugglers lure arab and african migrants by offering discounts to get onto overcrowded ships if people bring more potential passengers, a cnn investigation has revealed. (...)

Summary: cnn investigation **uncovers** the **business inside** a **human smuggling ring**.

Article: eyewitness video showing white north charleston police officer michael slager shooting to death an unarmed black man has exposed discrepancies in the reports of the first officers on the scene. (...)

Summary: more **questions than answers emerge** in **controversial s.c.** police shooting.

Bild aus See et al. (ACL 2017): *Get to the point! Summarization with pointer-generator networks*

- Paralleles Korpus (MT): Mehrsprachiges Korpus, das die **gleichen** Texte in mehreren Sprachen enthält

- 1 **Monitorkorpus:** offenes Korpus, dem ständig Texte hinzugefügt werden. Vorteile? Nachteile?
- 2 **Mehrsprachiges Korpus**
- 3 **Paralleles Korpus:** Mehrsprachiges Korpus, das die gleichen Texte in mehreren Sprachen enthält.

- 1 Korpora
- 2 Tokenisierung
 - Wörter
 - Satzsegmentierung
- 3 Wortnormalisierungen
- 4 Wortverteilungen
- 5 Appendix (Optional): NLTK

Wie viele Wörter hat der folgende Text?

It's a shame that our data-base is not up-to-date. It is a shame that um, data base A costs \$2300.50 and that database B costs \$5000. All databases cost far too much.

Tokenisierung

Tokenisierung ist ein Verarbeitungsschritt, der einen Eingabetext automatisch in Einheiten namens **tokens** segmentiert. Oft ist ein Token ein Wort, Zahl oder Interpunktion.

Wort: pragmatische Definition

Alphabetische Zeichensequenz, die von whitespace (space, tab, newline) abgegrenzt wird.

Wort: (eine) linguistische Definition

Kleinste Einheit einer Sprache, die allein stehen kann

Welche Probleme hat einfache White-Space-Tokenisierung?

- Punctuation wird am Ende des Vorgängerwortes belassen
- Punctuation und spezielle Zeichen in einem Token:
 - Abkürzungen: *Ph.D*, *Mr.*
 - andere Fälle: *\$2300.50*, *amazon.com*, *#Ironie?*
- Whitespaces in einem Token: *50 000*
- Clitics: *isn't?*
- Bindestriche: What about *data-base*, *database*, *data base*, *up-to-date*?
- Namen: *New York*, *rock 'n' roll*
- Komposita: *Donaudampfschiffahrtsgesellschaft*
- Sprachen wie Chinesisch benutzen keinen white space als Trennzeichen.
- Gesprochene Sprache: *um?*

Welche Probleme hat einfache White-Space-Tokenisierung?

- Punctuation wird am Ende des Vorgängerwortes belassen
- Punctuation und spezielle Zeichen in einem Token:
 - Abkürzungen: *Ph.D*, *Mr.*
 - andere Fälle: *\$2300.50*, *amazon.com*, *#Ironie?*
- Whitespaces in einem Token: 50 000
- Clitics: *isn't?*
- Bindestriche: What about *data-base*, *database*, *data base*, *up-to-date*?
- Namen: *New York*, *rock 'n' roll*
- Komposita: *Donaudampfschiffahrtsgesellschaft*
- Sprachen wie Chinesisch benutzen keinen white space als Trennzeichen.
- Gesprochene Sprache: *um?*

- Separiert Clitics und Anglo-Saxon Genitive
- Lässt Wörter mit Bindestrichen zusammen
- Separiert fast alle Punctuation

Beispiel (Underscore als Token Separator)

`"_We_think_that_the_San_Francisco-based_
restaurant_,_they_said_,_does_n't_charge_$_10_..`

Oft wird mit regulären Ausdrücken tokenisiert (schnell): Penn Treebank tokenizer

<https://web.archive.org/web/20151201051654/http://www.cis.upenn.edu/~treebank/tokenizer.sed>

Weiteres Beispiel für RE-Tokenizer

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)      # set flag to allow verbose regexps
...   ([A-Z]\.)+          # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?   # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [!,:;"'()?()-_']  # these are separate tokens; includes ], [
...   '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Bild aus Bird et al. (2009): *Natural Language Processing with Python*

Werden genrebedingt variieren (Twitter ...)

- Alternative: Neuronale Sequenzmodelle (z.B. für Chinesisch)
- Alternative: wordpiece Tokenisierer, der auf Language Modeling basiert
- **Alternative**: Finde häufige Symbolsequenzen (Byte-Pair encoding (BPE), SentencePiece encoding)

Grundideen

- Die Daten sagen uns, was gute Tokens sind, durch Vorkommenshäufigkeiten
- Tokens können kleiner als “whitespace Wörter” sein (Morpheme und subwords wie *iest* im Englischen oder *end* im Deutschen)

Vorteil: Ungesehene Wörter können aus *subwords* zusammengesetzt werden

- 1 Beginne mit einer Wortliste/dictionary bzw mit einer einfachen whitespace Tokenisierung, die einem das *dictionary* gibt
- 2 Dieses *dictionary* enthält auch initiale *word counts*
- 3 Initiales Vokabular V : alle Symbole/Buchstaben plus EOW-Symbol $_$
- 4 Zähle Symbolpaare. Füge das häufigste Symbolpaar (x, y) als ein neues zusammengefügt Symbol (xy) zum Vokabular hinzu und ersetze (x, y) im dictionary durch xy
- 5 Führe Schritt 4 k -mal aus
- 6 Finales Symbolset hat die Größe: $|V| + k$

Initiales *dictionary*:

```
5 low _
2 lowest _
6 newer _
3 wider _
2 new _
```

Initiales Vokabular: _, d, e, i, l, n, o, r, s, t, w

Häufigstes Symbolpaar: (r, _) 9mal

Ersetze im dictionary und füge zum Vokabular hinzu

Dictionary nach einer Iteration:

```
5  l o w _  
2  l o w e s t _  
6  n e w e r _  
3  w i d e r _  
2  n e w _
```

Vokabular nach einer Iteration: _, d, e, i, l, n, o, r, s, t, w, r_

Häufigstes Symbolpaar: (e, r_) (9mal)

Ersetze im dictionary und füge zum Vokabular hinzu

Dictionary nach zweiter Iteration:

```
5  l o w _  
2  l o w e s t _  
6  n e w e r _  
3  w i d e r _  
2  n e w _
```

Vokabular nach zweiter Iteration: _, d, e, i, l, n, o, r, s, t, w, r_, **er_**

Häufigstes Symbolpaar: (e, w) (8mal)

Ersetze im dictionary und füge zum Vokabular hinzu

Dictionary nach dritter Iteration:

5 l o w _
2 l o w e s t _
6 n ew er_
3 w i d e r_
2 n ew _

Vokabular nach dritter Iteration: _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew

Häufigstes Symbolpaar: (n, ew) (8mal)

Dictionary nach dritter Iteration:

5	l o w _
2	l o w e s t _
6	n ew er _
3	w i d er _
2	n ew _

Vokabular nach dritter Iteration: _, d, e, i, l, n, o, r, s, t, w, r_, er_, ew

Häufigstes Symbolpaar: (n, ew) (8mal)

Merge	Vokabular
(n, ew)	_, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, r_, er_, ew, new, lo, low, newer_, low_

BPE Testing: Tokenisierung neuer Sätze

Gegeben Testsatz s :

- 1 Tokenisiere s nach whitespace
- 2 Jedes white-space "Wort" in s wird in Buchstaben/Symbole zerlegt
- 3 Verwende dann die im Training gelernten Regeln, in der Reihenfolge, in der wir sie gelernt haben (häufigste Merges zuerst)

Neuer Testsatz/Testwort: *lower*

Unser Vokabular war $_, d, e, i, l, n, o, r, s, t, w, r_,$
 $er_, ew, new, lo, low, newer_, low_,$

- 1 Zerlege in Buchstaben $l\ o\ w\ e\ r\ _$
- 2 Merge $r_: l\ o\ w\ e\ r\ _$
- 3 Merge $er_: l\ o\ w\ e\ r\ _$
- 4 Merge $lo_: l\ o\ w\ e\ r\ _$
- 5 Merge $low_: l\ o\ w\ e\ r\ _$

BPE Testing: Tokenisierung neuer Sätze

Gegeben Testsatz s :

- 1 Tokenisiere s nach whitespace
- 2 Jedes white-space "Wort" in s wird in Buchstaben/Symbole zerlegt
- 3 Verwende dann die im Training gelernten Regeln, in der Reihenfolge, in der wir sie gelernt haben (häufigste Merges zuerst)

Neuer Testsatz/Testwort: *lower*

Unser Vokabular war $_, d, e, i, l, n, o, r, s, t, w, r_,$
 $er_, ew, new, lo, low, newer_, low_{}$

- 1 Zerlege in Buchstaben $l\ o\ w\ e\ r\ _$
- 2 Merge $r_$: $l\ o\ w\ e\ r\ _$
- 3 Merge $er_$: $l\ o\ w\ er\ _$
- 4 Merge lo : $lo\ w\ er\ _$
- 5 Merge low : $low\ er\ _$

- 1 Splittet white-space Wörter eventuell in subwords
- 2 Dies approximiert Morphologie (erlaubt Generalisierung in Testanwendungen)
- 3 Genreunabhängiger Algorithmus
- 4 In Wirklichkeit: sehr großes initiales *dictionary* und mehrere tausend merges
- 5 Die häufigen Wörter bleiben ganz
- 6 Seltene (und unbekannte) werden gesplittet
- 7 Keine Tokens über white space hinweg (*San Francisco*): Alternativen *SentencePiece* Modell (Kudos und Richardson, 2018).

Satz linguistisch

Eine grammatisch unabhängige/vollständige linguistische Einheit, die aus einem oder mehreren Wörtern bzw. Tokens besteht.

Satz naiv

Eine Folge von Wörtern, die in einem Punkt, Fragezeichen oder Ausrufezeichen endet

Aber

- Abkürzungen: Dr. Foster went to Glasgow.
- Indirekte Rede: He said "rubbish!".
- Satzzeichen wie –

Satz linguistisch

Eine grammatisch unabhängige/vollständige linguistische Einheit, die aus einem oder mehreren Wörtern bzw. Tokens besteht.

Satz naiv

Eine Folge von Wörtern, die in einem Punkt, Fragezeichen oder Ausrufezeichen endet

Aber

- Abkürzungen: Dr. Foster went to Glasgow.
- Indirekte Rede: He said "rubbish!".
- Satzzeichen wie –

Beispiel regelbasierte Satzsegmentierung: quick and dirty

IF current char is ? !

IF next char is not a quotation mark

split after current char

ELSE move to next char

ELSIF current char is .

IF preceded by abbr. that is not usually sentence final

move to next char

ELSIF preceded by abbr. and next-next char is not uppercase

move to next char

ELSIF next char is lower case or a number

move to next char

ELSE split after current char

ELSE move to next char

Alternativen:

- Regelbasiert: Reguläre Ausdrücke plus Abkürzungsverzeichnisse
- Methoden des Maschinellen Lernens
- Konsequenz aus Tokenisierung (z.B. in Stanford CoreNLP): Ein Satz ist zu Ende, wenn eine Punctuation wie (.,!,?) nicht schon in ein Token eingruppiert wurde.

- Korpora gibt es in verschiedensten Variationen → Textauswahl beeinflusst Anwendung
- Probleme bei Tokenisierung: white space sowie Punctuation genügt nicht
- Tokenisierung: kann regelbasiert (REs) oder durch statistische Methoden (BPE, wordpiece, sentencepiece) erfolgen

- ① * Jurafsky und Martin. Kapitel 2. Dritte online Edition.
- ② Für Suche nach Korpora:
<https://toolbox.google.com/datasetsearch> **oder**
<https://www.tensorflow.org/datasets>

- ❶ * Für BPE: Sennrich et al (ACL 2016): *Neural Machine Translation of rare words with subword units*
- ❷ * Gute Ressource für BPE-Vokabulare für 275 Sprachen:
<https://bpemb.h-its.org/>
- ❸ Bird et al (2009): NLTK Buch Kapitel 1 mit Übungen:
<https://www.nltk.org/book/>
- ❹ Stanford Core NLP:
<https://stanfordnlp.github.io/CoreNLP/> und Tokenizer
<http://nlp.stanford.edu/software/tokenizer.shtml>
- ❺ Für SentencePiece tokenizer: Kudo und Richardson (EMNLP 2018): *SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing*.
- ❻ Für word piece tokeniser: Devlin et al (NAACL 2019): *BERT: Pre-training of deep bidirectional transformers for language understanding*

- 1 Korpora
- 2 Tokenisierung
 - Wörter
 - Satzsegmentierung
- 3 Wortnormalisierungen**
- 4 Wortverteilungen
- 5 Appendix (Optional): NLTK

- **Lemma/Lexem**: Lexikonform eines Wortes (*Katze* für *Katzen*)
- **Wortform**: The voll inflektierte Oberflächenform eines Wortes, wie sie in einem Text vorkommt (*Katzen*, *Katze*)
- **(Primär)stamm/Wurzel**: Teil eines Wortes, der als Ausgangsbasis für weitere Wortbildung dienen kann. Oft (aber nicht immer) unvollständig. Als Primärstamm nicht weiter zerlegbar. Unflektiert.

Beispiel: *sicher* als Stamm von *sicherlich*, *sicherheit*, *sicherten*, *unsicher*, *unsicherheiten*

- Case folding: *The boy's mother is talking to the booksellers* → *the boy's mother is talking to the booksellers* wann? warum?
- Lemmatisierung: *the boy's mother is talking to the booksellers* → *the boy mother are talk to the bookseller*
- Stemming: einfachste Morphologie durch Affixentfernung (im Englischen)
the boy's mother is talking to the booksellers → *the boy mother is talk to the booksel*

Step 1a

sses	→ ss	caresses	→ caress
ies	→ i	ponies	→ poni
ss	→ ss	caress	→ caress
s	→ ∅	cats	→ cat

Step 1b

(*v*)ing	→ ∅	walking	→ walk
		sing	→ sing
(*v*)ed	→ ∅	plastered	→ plaster
...			

Step 2 (for long stems)

ational	→ ate	relational	→ relate
izer	→ ize	digitizer	→ digitize
ator	→ ate	operator	→ operate
...			

Step 3 (for longer stems)

al	→ ∅	revival	→ reviv
able	→ ∅	adjustable	→ adjust
ate	→ ∅	activate	→ activ
...			

- (context) $S1 \rightarrow S2$
- 5 Schritte: man geht jeden Schritt für jedes Wort durch und mehr als ein Schritt kann angewandt werden
- Innerhalb des Schrittes: nimm längste anwendbare S1 Seite
- Kontext: heuristische Bestimmung der Silbenanzahl, Vokale etc.

Annotation

Prozess, in dem Zusatzinformation zu einem Korpus hinzugefügt wird.
Erhöht die Nützlichkeit eines Korpus.

Annotationsmethoden

- Ground-Truth: von Experten (oder Autoren oder Crowdworkers oder ...) hinzugefügt
- Automatisch: durch NLP Algorithmen hinzugefügt

Metadaten meist in einem `text/corpus` header

- Erstellungsdatum
- Author/Sprecher
- Copyright
- Titel
- Text Genre
-

TEI (Text Encoding Initiative) publiziert Corpus Encoding Standards

<http://www.cs.vassar.edu/CES/>

- Linguistische Annotation: Wortgrenzen, Satzgrenzen, POS, Eigennamen, Syntax ...
- Relationen zwischen verschiedenen Annotationen möglich
- Anderes: Meinungen, Links zu Wissensbasen ...
- In HTML/XML. Beispiel:

```
<w id="bncCPB-s276-w1" type="AJ0"> Creative </w>  
<w id="bncCPB-s276-w2" type="NN2"> Technologies </w>  
<w id="bncCPB-s276-w3" type="AJ0"> Inc </w>  
<w id="bncCPB-s276-w4" type="VHZ"> has </w>  
<w id="bncCPB-s276-w5" type="VVN"> reported </w>
```

- Oft in Standoff-Format

- Außer Tokenisierung können noch weitergehende Normalisierungsschritte erfolgen (Case Folding, Stemming, Lemmatisierung)
- Korpora können “roh” sein, nur mit Metadaten, oder weiter annotiert (Tokenisiert, POS-tagged, Syntaxbäume, ...)
- *Porter, Martin (1980): An algorithm for suffix stripping. *Program*, 24(3), 130-137.
<https://tartarus.org/martin/PorterStemmer/def.txt>

- 1 Korpora
- 2 Tokenisierung
 - Wörter
 - Satzsegmentierung
- 3 Wortnormalisierungen
- 4 Wortverteilungen
- 5 Appendix (Optional): NLTK

Tokens

Ein Worttoken ist ein individuelles Vorkommen eines Wortes. Wie groß ist das Korpus (N)? = wie viele **word tokens** gibt es?

Types

Wie viele verschiedene Wörter (**word types**) gibt es? Gibt das Korpusvokabular (V) an.

Type-token Verteilung

Was ist die Frequenz jedes **word type**?

Lemma/Lexem

Lemma/Lexem: Lexikonform eines Wortes. *kosteten* und *koste* kommen vom gleichen Lemma *kosten*.

Tokens

Ein Worttoken ist ein individuelles Vorkommen eines Wortes. Wie groß ist das Korpus (N)? = wie viele **word tokens** gibt es?

Types

Wie viele verschiedene Wörter (**word types**) gibt es? Gibt das Korpusvokabular (V) an.

Type-token Verteilung

Was ist die Frequenz jedes **word type**?

Lemma/Lexem

Lemma/Lexem: Lexikonform eines Wortes. *kosteten* und *koste* kommen vom gleichen Lemma *kosten*.

Tokens

Ein Worttoken ist ein individuelles Vorkommen eines Wortes. Wie groß ist das Korpus (N)? = wie viele **word tokens** gibt es?

Types

Wie viele verschiedene Wörter (**word types**) gibt es? Gibt das Korpusvokabular (V) an.

Type-token Verteilung

Was ist die Frequenz jedes **word type**?

Lemma/Lexem

Lemma/Lexem: Lexikonform eines Wortes. *kosteten* und *koste* kommen vom gleichen Lemma *kosten*.

Tokens

Ein Worttoken ist ein individuelles Vorkommen eines Wortes. Wie groß ist das Korpus (N)? = wie viele **word tokens** gibt es?

Types

Wie viele verschiedene Wörter (**word types**) gibt es? Gibt das Korpusvokabular (V) an.

Type-token Verteilung

Was ist die Frequenz jedes **word type**?

Lemma/Lexem

Lemma/Lexem: Lexikonform eines Wortes. *kosteten* und *koste* kommen vom gleichen Lemma *kosten*.

- **Input:** Text
- **Output:** Liste von Wörtern mit Häufigkeit
- **Algorithmus:**
 - Tokenize (`tr`)
 - Sort (`sort`)
 - Zähle Duplikate (`uniq -c`)

Der Befehl `tr` substituiert Zeichen.

```
tr 'chars1' 'chars2' < inputfile > outputfile
```

Beispiel: Was macht?

```
tr '[a-z]' '[A-Z]' < case
```

Input:

“My dear fellow.” said Sherlock Holmes as we sat on either side of the fire in his lodgings at Baker Street, “ life is infinitely stranger than anything which the mind of man could invent.

```
tr ' [a-z]' ' [A-Z]' < case
```

Output:

"MY DEAR FELLOW." SAID SHERLOCK HOLMES AS WE SAT
ON EITHER SIDE OF THE FIRE IN HIS LODGINGS AT BAKER
STREET, "LIFE IS INFINITELY STRANGER THAN ANYTHING
WHICH THE MIND OF MAN COULD INVENT.

```
tr '[A-Za-z]' '\012' < case
```

verwandelt alphabetische Zeichen in newlines

```
tr -c '[A-Za-z]' '\012' < case
```

tut das Gegenteil

```
My
dear
fellow
```

```
said
Sherlock
```

```
tr '[A-Za-z]' '\012' < case
```

verwandelt alphabetische Zeichen in newlines

```
tr -c '[A-Za-z]' '\012' < case
```

tut das Gegenteil

```
My  
dear  
fellow
```

```
said  
Sherlock
```

```
tr '[A-Za-z]' '\012' < case
```

verwandelt alphabetische Zeichen in newlines

```
tr -c '[A-Za-z]' '\012' < case
```

tut das Gegenteil

My
dear
fellow

said
Sherlock

```
tr -cs '[A-Za-z]' '\012' < case
```

entfernt newline Wiederholungen

My

dear

fellow

said

Sherlock

Holmes

as

we

sat

```
tr -cs '[A-Za-z]' '\012' < case | sort
```

sortiert alphabetisch

a

a

....

A

A

...

able

able

...

about

about

```
tr -cs '[A-Za-z]' '\012' < case | sort | uniq -c
```

entfernt Duplikate, aber zählt sie vorher!

```
158  a
7    A
4    able
13   about
1    About
46   Holmes
```

```
tr -cs '[A-Za-z]' '\012' < case | sort | uniq -c
```

entfernt Duplikate, aber zählt sie vorher!

158	a
7	A
4	able
13	about
1	About
46	Holmes

Sortiere nach Häufigkeit

```
tr -cs '[A-Za-z]' '\012' < case |  
sort | uniq -c | sort -nr
```

330	the	
194	and	47 which
190	to	47 have
163	of	47 but
160	I	46 me
158	a	46 Holmes
128	that	20 The

Was machen die folgenden Befehle?

```
tr '[A-Z]' '[a-z]' < case |  
tr -cs '[a-z]' '\012' |  
sort |  
uniq -c |  
sort -nr
```

Was machen die folgenden Befehle?

<code>tr ' [A-Z]' ' [a-z]' < case </code>	<code># Case Folding</code>
<code>tr -cs ' [a-z]' ' \012' </code>	<code># tokenisieren</code>
<code>sort </code>	<code># alphabetisch sortieren</code>
<code>uniq -c </code>	<code># entduplizieren und zählen</code>
<code>sort -nr</code>	<code># numerisch absteigend sortieren</code>

Jetzt		Vorher	
350	the	330	the
212	and	194	and
191	to	190	to
167	of	163	of
165	a	160	I
160	i	158	a

Meist zählt man nach Case Folding!

Wie viele Worttokens enthält “A Case of Identity”?

```
tr '[A-Z]' '[a-z]' < case | tr -cs '[a-z]' '\012' | wc  
-l    7105
```

Wie viele verschiedene Wörter (Types) enthält “A case of identity”?

```
tr '[A-Z]' '[a-z]' < case | tr -cs '[a-z]' '\012' |  
sort | uniq | wc -l    1625
```

Wie viele Worttokens enthält “A Case of Identity”?

```
tr '[A-Z]' '[a-z]' < case | tr -cs '[a-z]' '\012' | wc  
-l    7105
```

Wie viele verschiedene Wörter (Types) enthält “A case of identity”?

```
tr '[A-Z]' '[a-z]' < case | tr -cs '[a-z]' '\012' |  
sort | uniq | wc -l    1625
```

Corpus	Tokens N	Types V
Case of Identity	7105 (10^3)	1625 (10^3)
Switchboard	2.4m (10^6)	20 000 (10^4)
BNC	100m (10^8)	636 397 (10^5)
Web 1T-gram	1B (10^{12})	13m (10^7)

Heaps Law

$$|V| = KN^\beta, \text{ mit } K \text{ positiv, } 0 < \beta < 1$$

Je nach Korpus:

- K zwischen 10 und 100
- β zwischen 0.4 und 0.6

Sherlock Holmes Story: A case of identity.

“My dear fellow.” said Sherlock Holmes as we sat on either side of the fire in his lodgings at Baker Street, “ life is infinitely stranger than anything which the mind of man could invent.

Resultat

350 the

212 and

191 to

167 of

165 a

160 i

Für "A Case of Identity"

Wort Häufigkeit	Häufigkeit von Häufigkeit
1	993
2	248
3	93
4	70
5	40
10	8
50	2
>100	11

7105 Tokens

1625 Types

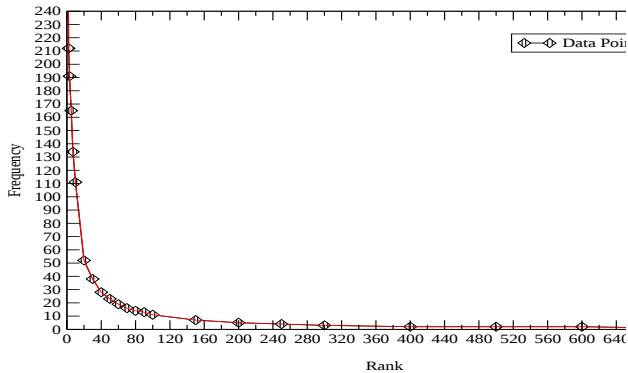
Zipfssches Gesetz (Zipf's law) I

Wort	Häuf. (f)	Rang (r)	$f \cdot r$
the	350	1	350
and	212	2	424
to	191	3	573
was	111	10	1110
her	52	20	1040
had	38	30	1140
very	28	40	1120
what	23	50	1150
father	19	60	1140
come	16	70	1120

Zipfssches Gesetz (Zipf's law) I

Wort	Häuf. (f)	Rang (r)	$f \cdot r$
out	14	80	1120
can	13	90	1170
street	11	100	1100
time	7	150	1050
leadenhall	5	200	1000
went	3	300	900
violent	2	400	800
mean	2	500	1000
certain	2	600	1200
unprof	1	700	700
pleasant	1	1625	1625

Häufigkeit und Rang



Häufigkeit lässt sich durch Rang/Position abschätzen. (Harvard Linguist George Kingsley Zipf).

Es gibt Konstante k so dass:

$$f \cdot r = k$$

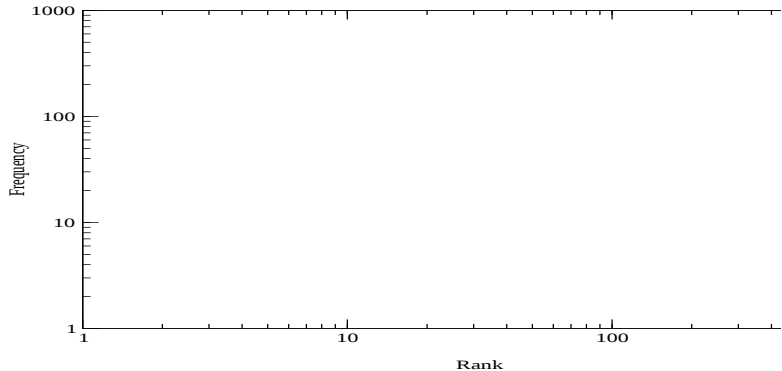
Oder f ist **power-law Funktion von** r (Hyperbel):

$$f \propto \frac{1}{r}$$

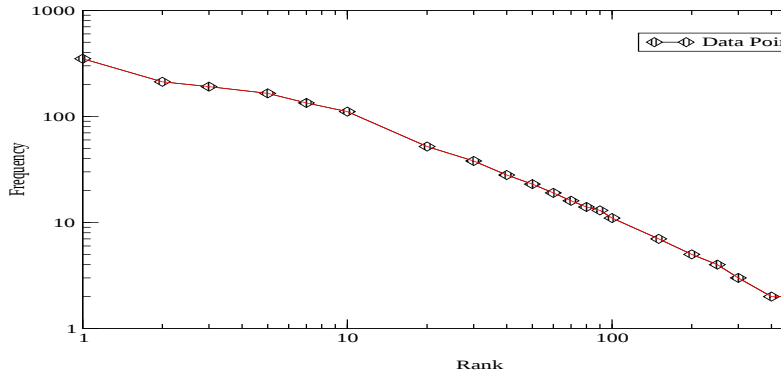
bzw. im Logarithmus

$$\log f = \log k - 1 \cdot \log r$$

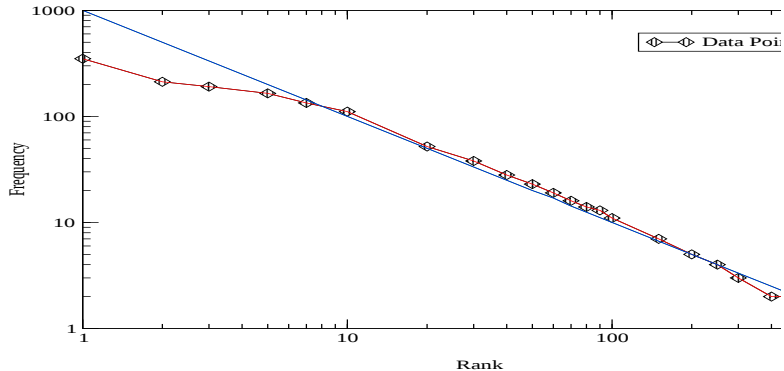
Logarithmische Skala



Logarithmische Skala



Logarithmische Skala



- Sehr kleine Anzahl von häufigen Wörtern
- Relativ kleine Anzahl von mittelhäufigen Wörtern
- Sehr große Anzahl von seltenen Wörtern
- Beziehung zwischen Häufigkeit und Rang kann durch Gerade (in logarithmischer Skala) angenähert werden
- Nicht so schöne Schätzeigenschaften wie Gaußverteilung

Welche anderen Phänomene werden durch Zipfsche Verteilung modelliert?

Welche Phänomene werden durch Gauß-Verteilungen modelliert?

Zipf für das Brown Korpus

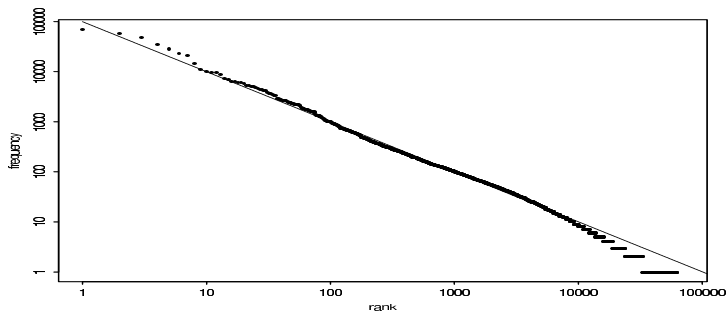


Bild aus Manning und Schütze (1999): Foundations of Statistical Natural Language Processing. Figure 1.1

Mandelbrot verfeinerte Zipfsches Gesetz (besserer fit)

$$f = k(r + \beta)^{-\alpha} \quad \text{oder} \quad \log f = \log k - \alpha \log(r + \beta)$$

- Passt besser für sehr niedrige und sehr hohe Ränge
- k, β, α parametrisiert für bestimmte Korpora
- $\alpha = 1$ and $\beta = 0$?
- $\alpha = 1, \beta = 1$?
- Baroni (2005): α nah an 1 für alle untersuchten Korpora (zwischen 1.04 und 1.09),

- Type-Token-Verteilungen gehorchen Heaps Law
- Tokenverteilungen folgen Zipf's Law: Zusammenhang zwischen Häufigkeit und Rang
- Schätzung kann unter ungünstigen Verteilungen leiden
- **Data Sparseness:** Für die meisten Worte haben wir keine oder sehr geringe Evidenz

- ❶ * Jurafsky und Martin. Kapitel 2. Dritte online Edition.
- ❷ Church, Ken (1994). *Unix for Poets*.
<https://www.cs.upc.edu/~padro/Unixforpoets.pdf>
Herunterladbar auf Modulseite
- ❸ Bird et al (2009): NLTK Buch Kapitel 1 mit Übungen:
<https://www.nltk.org/book/>
- ❹ Heaps, Harold Stanley (1978): *Information retrieval. Computational and theoretical aspects*. Academic Press
- ❺ Zipf, GK (1949). *Human Behavior and the Principle of Least Effort*.
- ❻ Sehr empfehlenswert: Baroni, M. (2005): 39 distributions in text.
In *Corpus Linguistics: An international handbook*. https://marcobaroni.org/publications/hsk_39_dist_rev2.pdf