Korpora, Tokenisierung, Normalisierung, Wortverteilungen

Katja Markert

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

November 10, 2019

Übersicht

Bisher und jetzt

- Zeichenketten
- Erkennung, Generierung, Stringvergleich
- Jetzt: Wie teile ich einen Text überhaupt in bestimmte Zeichenketten wie Wörter und Sätze ein? Wie normalisiere ich Text?
- Jetzt: Für empirische NLP und zum Lernen aus Texten braucht man Textsammlungen. Was für Textsammlungen gibt es?

Übersicht

Was ist ein Korpus?

das Korpus: (heutige) Definition

endliche Sammlung von maschinenlesbaren, natürlich vorkommenden Texten; oft nach bestimmten Kriterien selektiert

Man lernt verschiedene Dinge aus verschiedenen Texten!

Beispiele:

- Wortbedeutungen
- Wortsequenzen und Grammatik
- Textstruktur

Korpusunterschiede

- Sprachtyp: welche Sprache(n) von 7,111 lebenden Sprachen (`www.ethnologue.com`), Dialekte vs. “standard”
- Genre und Domänen: Romane, Zeitungstexte, Dialoge....
- Zeit und Autor/Sprecher
- Medium: Textuell, Audio, Transkriptionen, Video.
- Größe

Korpusbeispiele I

*A*nd 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?

Korpusbeispiele II

”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?

Wichtige Korpora

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

Links (die meisten auch auf ella unter resources z.b. `/resources/corpora/monolingual/raw/gigaword_eng_5`):

- Google n-grams: `http://www.ldc.upenn.edu/`
- BNC: `http://www.natcorp.ox.ac.uk/`. Besseres Interface unter `http://bncweb.lancs.ac.uk/bncwebSignup/user/login.php`
- Brown corpus: part of nltk `http://www.nltk.org`
- DWDS Korpora: `www.dwds.de`
- Viele Korpora für bestimmte Anwendungen, z.B. DUC `https://www-nlpir.nist.gov/projects/duc/data.html` für automatische Textzusammenfassung

Übersicht

Was ist ein Wort I?

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/Tokenization

Tokenisierung

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

Wort: pragmatische und (falsche) Definition

Alphabetische Zeichensequenz, die von whitespace (space, tab, newline) abgegrenzt wird. Kann mit einem regulärem Ausdruck gemacht werden.

Wörter zählen mit Unix

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

Der Befehl `tr` I

DerBefehl `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.

Der Befehl `tr` II

```
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.

Der Befehl `tr` III: Tokenisierung

```
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

Der Befehl `tr` IV: Tokenisierung

```
tr -cs '[A-Za-z]' '\012' < case
# entfernt newline Wiederholungen
My
dear
fellow
said
Sherlock
Holmes
as
we
sat
```

Der Befehl `sort`

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

a
a
....
A
A
...
able
able
...
about
about
```

Der Befehl `uniq`

```
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

Mehr Befehle

Was machen die folgenden Befehle?

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

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

Mehr Befehle

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!

Welche Anwendungen brauchen Case Folding, welche nicht?

Was ist ein Wort II?

Welche Probleme hat diese einfache Tokenisierung, die wir benutzt haben?

- Punktuation braucht man meist noch $\longrightarrow$ parsing.
- Punktuation und spezielle Zeichen in einem Token:
  - Abkürzungen: *Ph.D*
  - 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: *Donaudampfschifffahrtsgesellschaft*
- Sprachen wie Chinesisch benutzen keinen white space als Trennzeichen.
- Gesprochene Sprache: *um*?

Penn Treebank Tokenization Standard

- Separiert Clitics
- Lässt Wörter mit Bindestrichen zusammen
- Separiert alle Punktuation

Beispiel (Underscore als Token Separator)

“_We_think_that_the_San_Francisco-based_ restaurant_,_“_they_said_,_“_does_n’t_charge_$_10_“_.

Heutige Tokenizer

1. Oft wird mit regulären Ausdrücken tokenisiert (schnell, deterministisch)
2. Alternative: Neuronale Sequenzmodelle (z.B. für Chinesisch)
3. Alternative: Finde häufige Symbolsequenzen (Byte-Pair encoding)
4. Alternative: wordpiece Tokenisierer, der auf Language Modeling basiert

Sie werden im Übungsblatt mit regulären Ausdrücken normalisieren.

Definitionen

Tokens

Ein Worttoken ist ein indiduelles 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*.

Korpusgröße

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
```

Type-Token Verteilungen

Corpus	Tokens N	Types $\|V\|$
Case of Identity	7105	1625
BNC	100m	636 397
Switchboard	2.4m	20 000
Google N-Grams	1B	13m

Heaps Law

$$|V| = kN^{\beta}, \textit{mit } k \textit{ positiv}, \ 0 < \beta < 1$$

Normalisierung

- Case folding: wann? warum?
- Lemmatisierung: *the boy's mother is reading a book* $\longrightarrow$ *the boy mother are read a book*
- Stemming: einfachste Morphologie durch Affixentfernung

for example compressed and compression are both accepted as equivalent to compress.

for exampl compress and compress ar both accept as equival to compress

Porter Stemmer

Step 1a

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

Step 1b

(*v*)ing	→ ø	walking	→ walk
		sing	→ sing
(*v*)ed	→ ø	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	→ ø	revival	→ reviv
able	→ ø	adjustable	→ adjust
ate	→ ø	activate	→ activ

…

Was ist ein Satz?

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 –

Heutige Methoden für Satzsegmentierung

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 Punktuation wie (.,!,?) nicht schon in ein Token eingruppiert wurde.

NLTK

Website: `nltk.org`.

- Inkludiert schon viele (tokenisierte) Korpora.
- Tools.
- Erlaubt eigene Tools für alte und neue Texte mit Python zu bauen.

NLTK

```
python
>>>import nltk
>>> nltk.download()
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

Tokenisierte Texte in NLTK sind Wortlisten

Diese können indiziert und gezählt werden.

```
>>>>text3[0:100]
>>>len(text3)
44764
>>>sorted(set(text3))
'!', "'", '(', ')', ',', ',)', '.', '.)', ':', ';', ';)', '?
'A', 'Abel', 'Abelmizraim', 'Abidah', 'Abide', 'Abimael',
'Abimelech','Abr', 'Abrah', 'Abraham', 'Abram', 'Accad',
'Achbor', 'Adah', ...]
>>> len(set(text3))
2789
```

Worthäufigkeiten zählen

- Häufigkeit eines Worttyps zählen

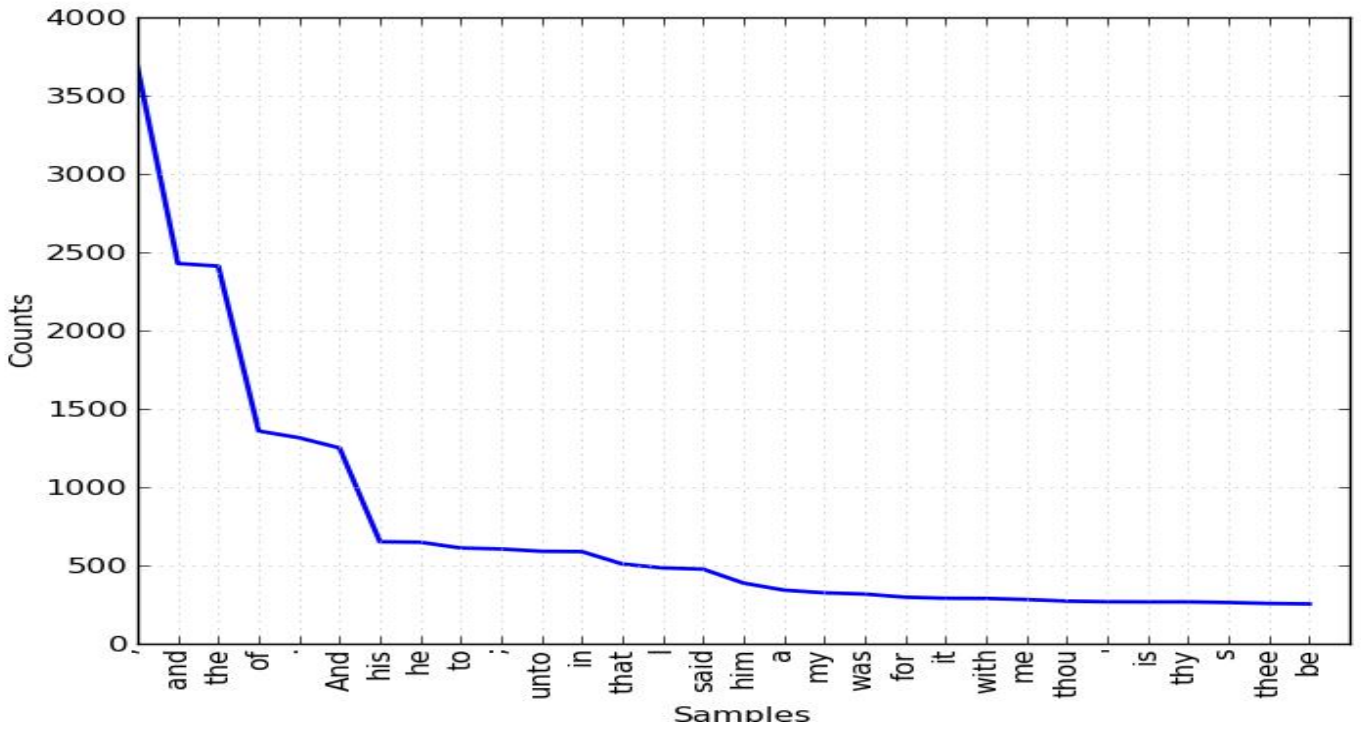
```
>>>text3.count("smote")
5
```

- Häufigkeit aller Worttypen zählen

```
>>>fdist1 = FreqDist(text3)
>>>vocabulary1 =fdist1.keys()
>>> vocabulary1[:30]
[',', 'and', 'the', 'of', '.', 'And', 'his', 'he', 'to',
';', 'unto', 'in', 'that', 'I', 'said', 'him', 'a',
'my', 'was', 'for', 'it', 'with', 'me', 'thou', "'", 'is
'thy', 's', 'thee', 'be']
>>>fdist1['thou']
272
```

Darstellung von Wortfrequenzen

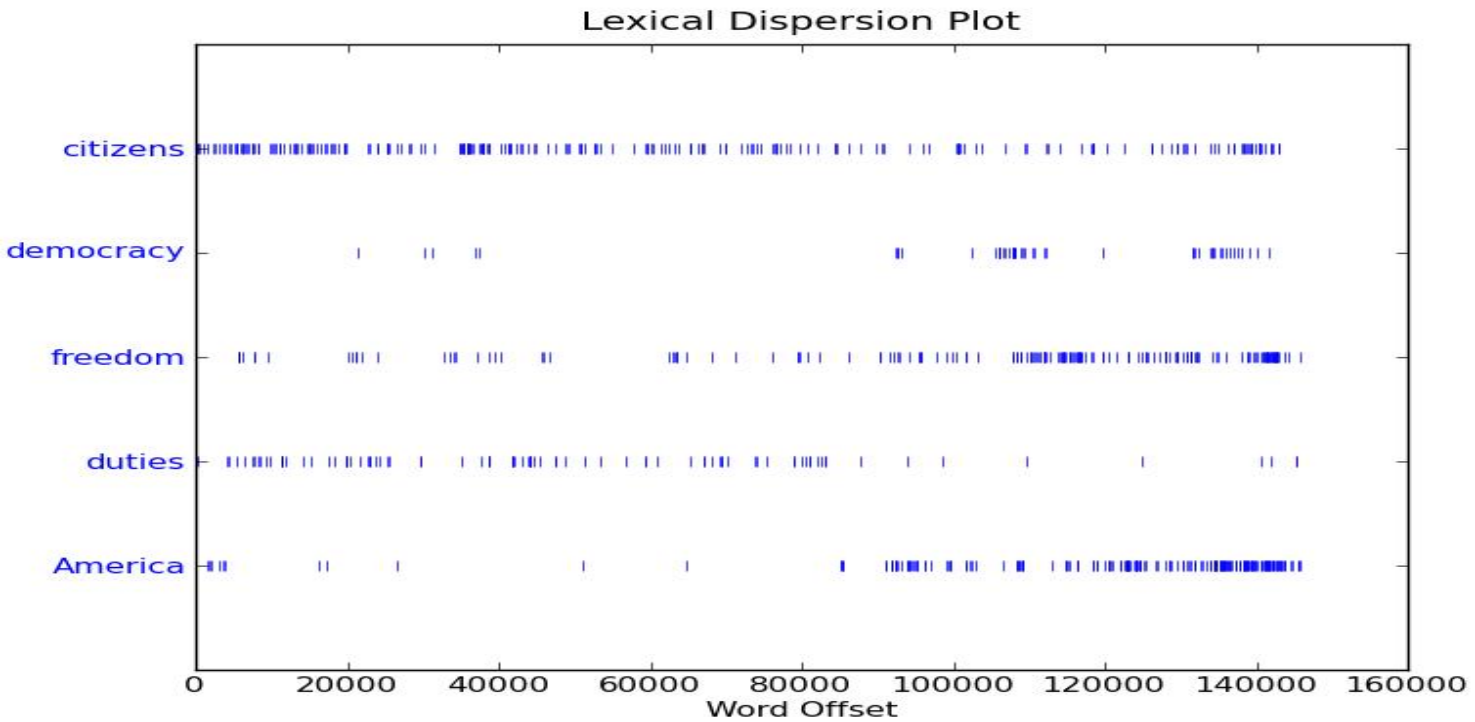
```
>>>fdist1.plot(30)
```



Wortverteilungen vergleichen

Text 4 Antrittsreden aller US Präsidenten

```
>>>text4.dispersion_plot(["citizens", "democracy", "freedom", "duties", "
```



Zwischenfazit

- Korpora gibt es in verschiedensten Variationen.
- Einfache Unixmethoden zur Tokenisierung und Wortzählung
- Probleme bei Tokenisierung, Normalisierung, Satzsegmentierung

Übersicht

Wörter zählen. Wir erinnern uns:

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

Häufigkeiten von Häufigkeiten

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

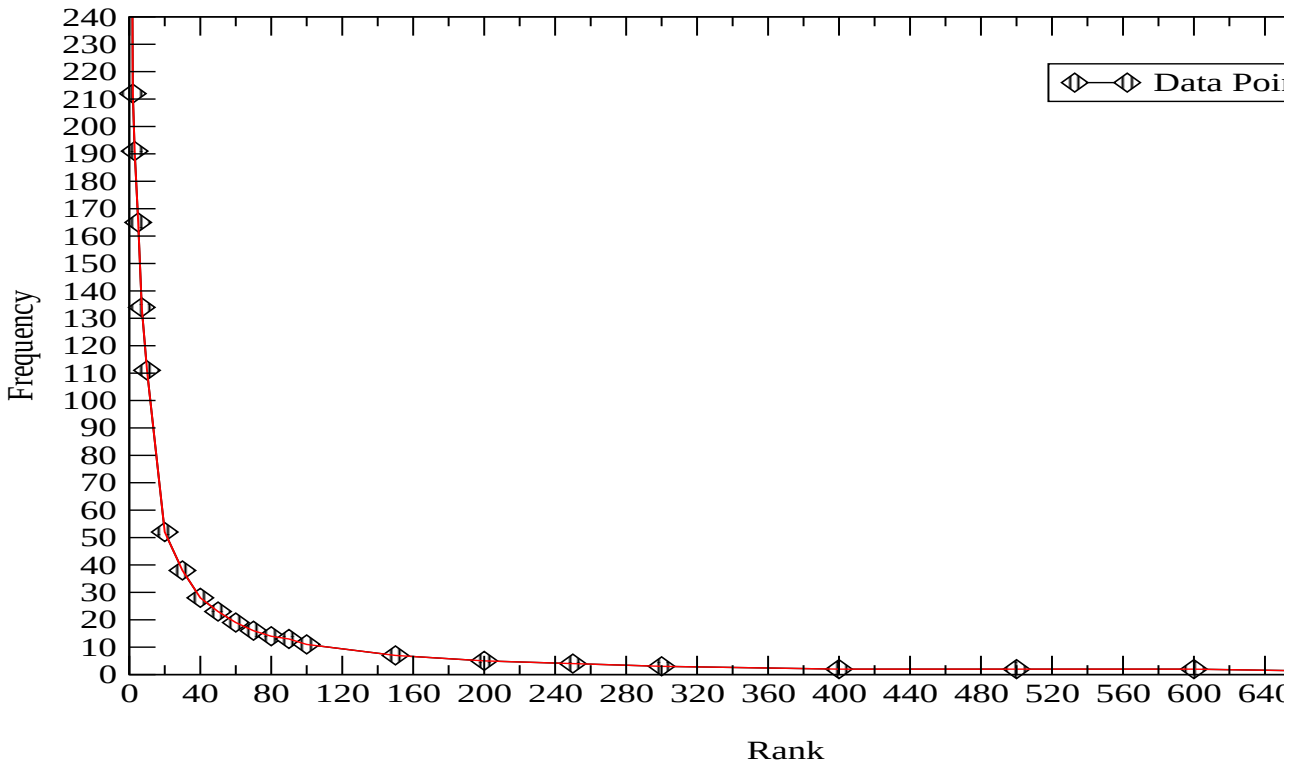
7105	Tokens
1625	Types

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

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

Häufigkeit und Rang



Zipfssches Gesetz II

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

Es gibt konstante K so dass:

$$f \cdot r = k$$

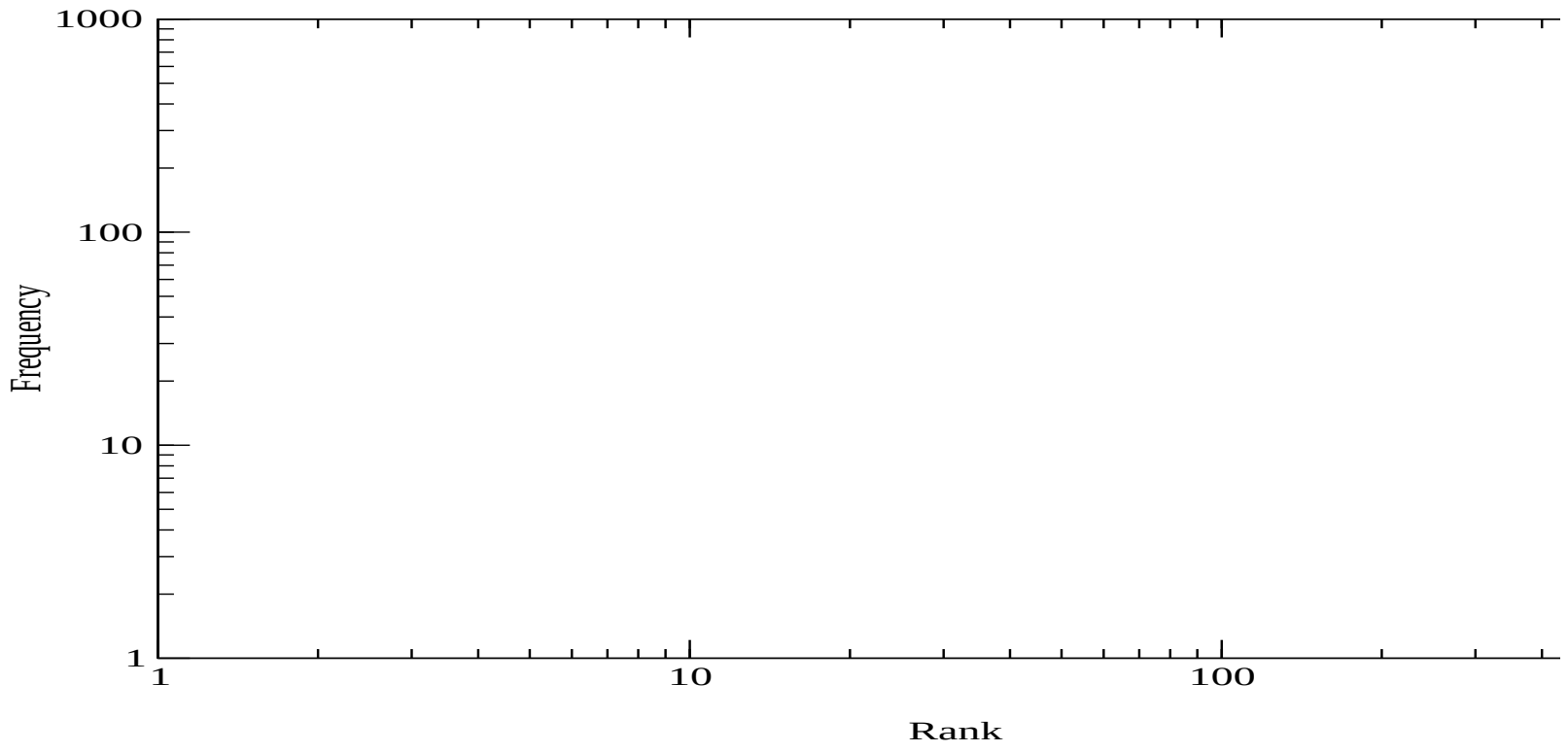
Oder f ist **power-law Funktion von** r (Hyperbel):

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

Mandelbrot verfeinerte Zipfsches Gesetz

$$f = k(r+p)^{-B} \quad \text{or} \quad \log f = \log k - B \log(r+p)$$

Logarithmische Skala



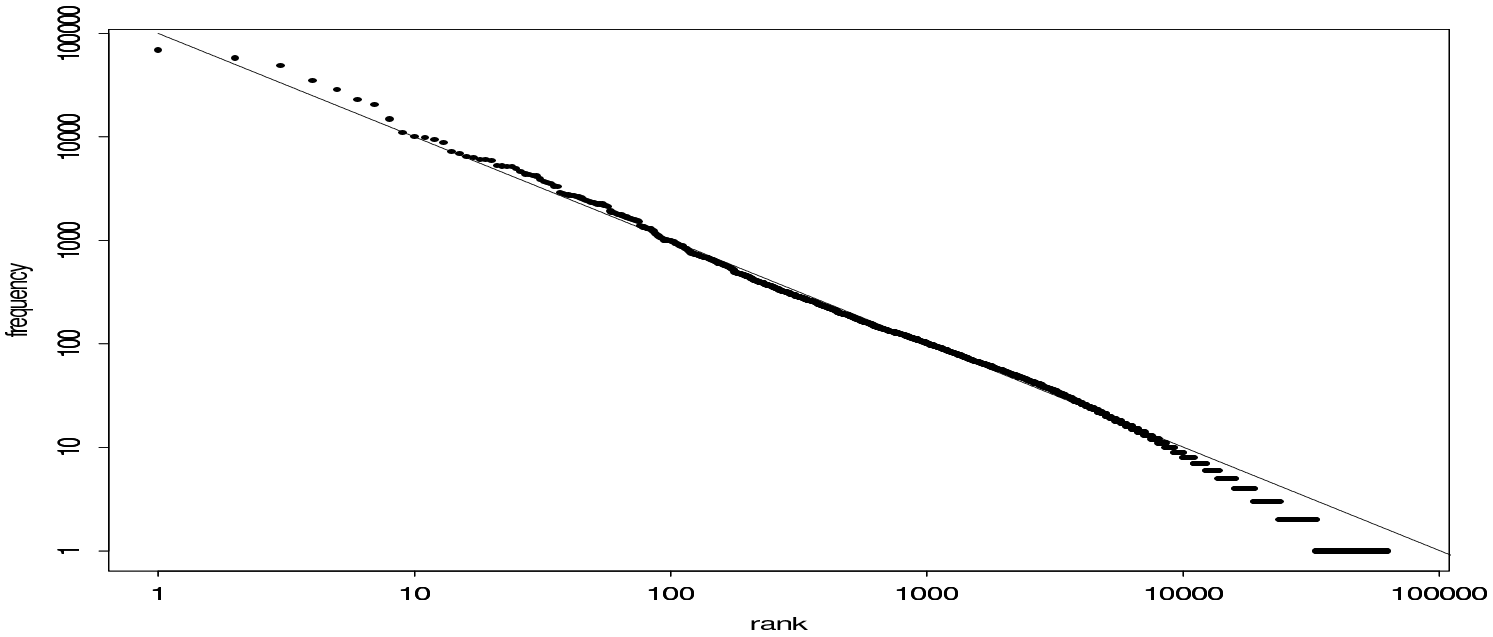
Zipfsches Gesetz: Konsequenzen

- 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



Zusammenfassung

- Schätzung kann unter ungünstigen Verteilungen leiden
- **Data Sparseness:** Für die meisten Worte haben wir keine oder sehr geringe Evidenz
- Zipfsches Gesetz: Zusammenhang zwischen Häufigkeit und Rang

Referenzen

1. * Jurafsky und Martin. Kapitel 2.2 - 2.4. Dritte online Edition.
2. Ken Church (1994). *Unix for Poets*.
   `https://www.cs.upc.edu/~padro/Unixforpoets.pdf`
   Herunterladbar auf Modulseite
3. Bird et al (2009): NLTK Buch Kapitel 1 mit Übungen:
   urlhttps://www.nltk.org/book/
4. Stanford Core NLP:
   `https://stanfordnlp.github.io/CoreNLP/`
5. * Porter (1980): An algorithm for suffix stripping. *Program, 24(3)*, 130-137.
6. Für Suche nach Korpora:
   `https://toolbox.google.com/datasetsearch`