Information Storage and Retrieval

CSCE 670
Texas A&M University
Department of Computer Science & Engineering
Instructor: Prof. James Caverlee

Statistical Properties of Text
+ Ranked Retrieval
24 January 2017

First off …

- Some statistical properties of text
  - Heaps' law
  - Zipf's law

Vocabulary vs Collection Size

- How big is the term vocabulary?
  - i.e., how many distinct words are there?
- Can we assume an upper bound?
- In practice, the vocabulary will keep growing with the collection size

Vocabulary Growth: Heaps' Law

- How does the size of the overall vocabulary (number of unique words) grow with the size of the corpus?
  - Vocabulary has no upper bound due to proper names, typos, etc.
  - New words occur less frequently as vocabulary grows

How big is the vocabulary?

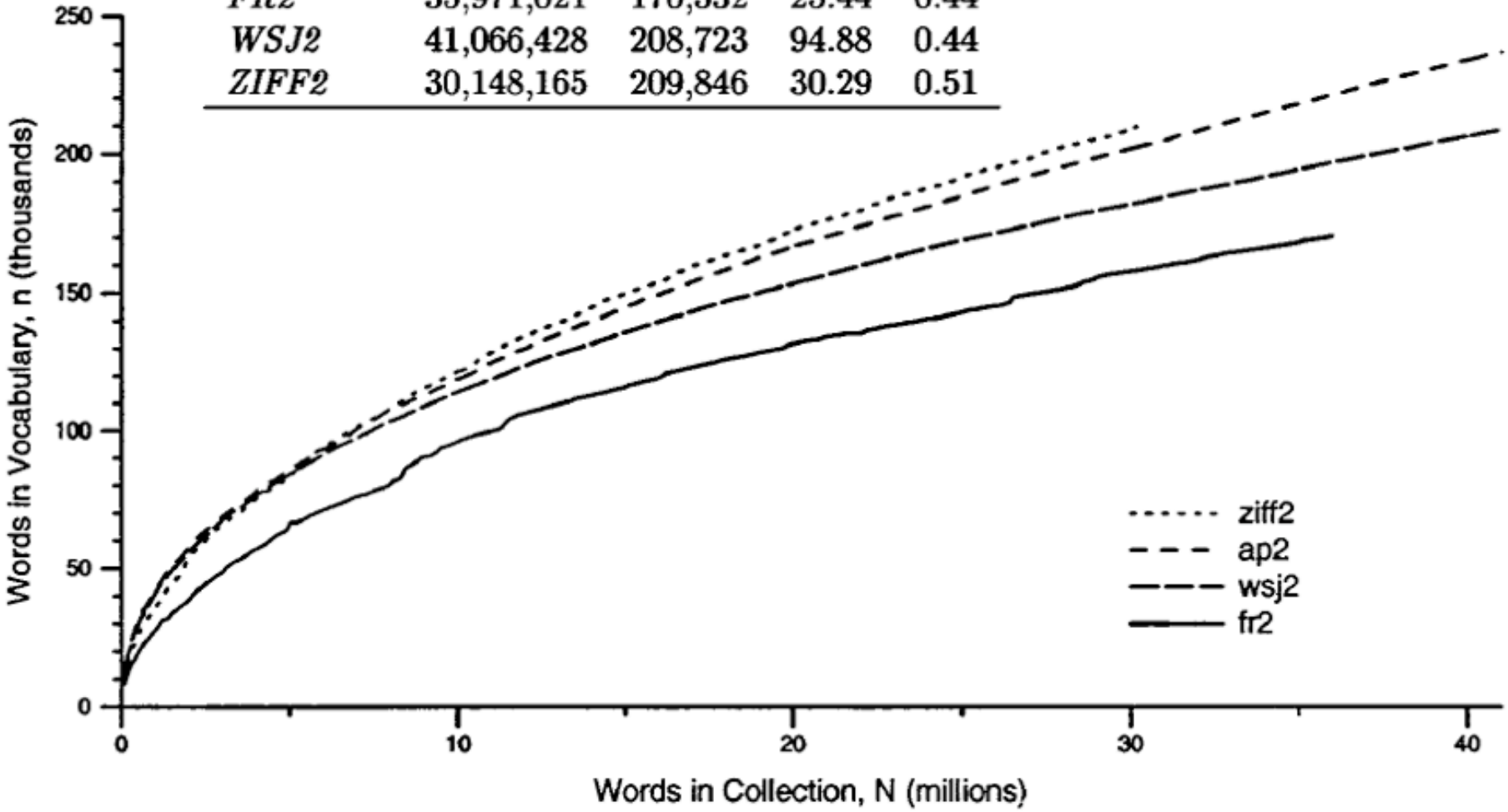
- Grows (but more slowly) with corpus size
- Empirically okay model: Heap's Law

$$m = kT^b$$

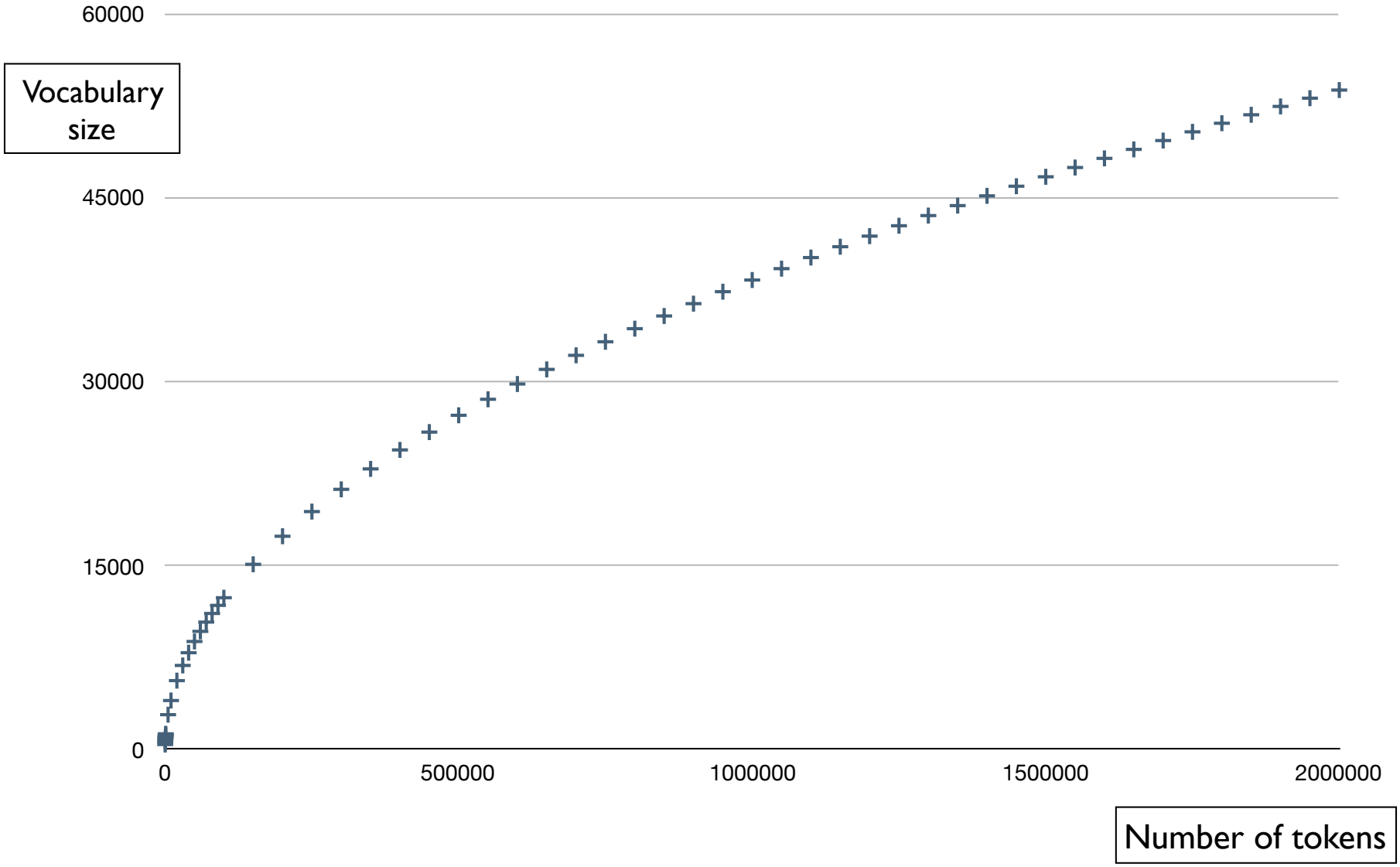
- where $b \approx 0.5$, $k \approx 30$–100; T = # tokens; m = size of the vocabulary
- For instance TREC disks 1 and 2 (2 GB; 750,000 newswire articles): ≈ 500,000 terms
- m is decreased by case-folding, stemming
- Indexing all numbers could make it extremely large (so usually don't)
- Spelling errors contribute a fair bit of size

$V = Kn^{\beta}$

Collection	N	n	K	β
AP2	40,998,865	237,160	63.11	0.47
FR2	35,971,021	170,532	25.44	0.44
WSJ2	41,066,428	208,723	94.88	0.44
ZIFF2	30,148,165	209,846	30.29	0.51



Heaps' law for Reuters (b=0.49, k=44)



Zipf's law

- The *i*th most frequent term has frequency proportional to *1/i*.
- cf is the collection frequency: the number of occurrences of the term in the collection
- A few words occur very often
- Many words are infrequent
- Zipf, 1902-1950: linguistic prof at Harvard

$$\mathrm{cf}_i \propto \frac{1}{i}$$

Word	Freq	r	Pr	r*Pr
the	15659	1	6.422	0.0642
of	7179	2	2.944	0.0589
to	6287	3	2.578	0.0774
a	5830	4	2.391	0.0956
and	5580	5	2.288	0.1144
in	5245	6	2.151	0.1291
that	2494	7	1.023	0.0716
for	2197	8	0.901	0.0721
was	2147	9	0.881	0.0792
with	1824	10	0.748	0.0748
his	1813	11	0.744	0.0818
is	1800	12	0.738	0.0886
he	1687	13	0.692	0.0899
as	1576	14	0.646	0.0905
on	1523	15	0.625	0.0937
by	1443	16	0.592	0.0947
at	1318	17	0.541	0.0919
it	1232	18	0.505	0.0909
from	1217	19	0.499	0.0948
but	1136	20	0.466	0.0932
u	949	21	0.389	0.0817
had	937	22	0.384	0.0845
last	909	23	0.373	0.0857
be	906	24	0.372	0.0892
who	883	25	0.362	0.0905

Word	Freq	r	Pr	r*Pr
has	880	26	0.361	0.0938
not	875	27	0.359	0.0969
an	863	28	0.354	0.0991
s	862	29	0.354	0.1025
have	860	30	0.353	0.1058
were	858	31	0.352	0.1091
their	812	32	0.333	0.1066
are	807	33	0.331	0.1092
one	742	34	0.304	0.1035
they	679	35	0.278	0.0975
its	668	36	0.274	0.0986
all	646	37	0.265	0.098
week	626	38	0.257	0.0976
government	582	39	0.239	0.0931
when	577	40	0.237	0.0947
would	572	41	0.235	0.0962
been	554	42	0.227	0.0954
out	553	43	0.227	0.0975
new	544	44	0.223	0.0982
which	539	45	0.221	0.0995
up	539	45	0.221	0.0995
more	535	47	0.219	0.1031
into	516	48	0.212	0.1016
only	504	49	0.207	0.1013
will	488	50	0.2	0.1001

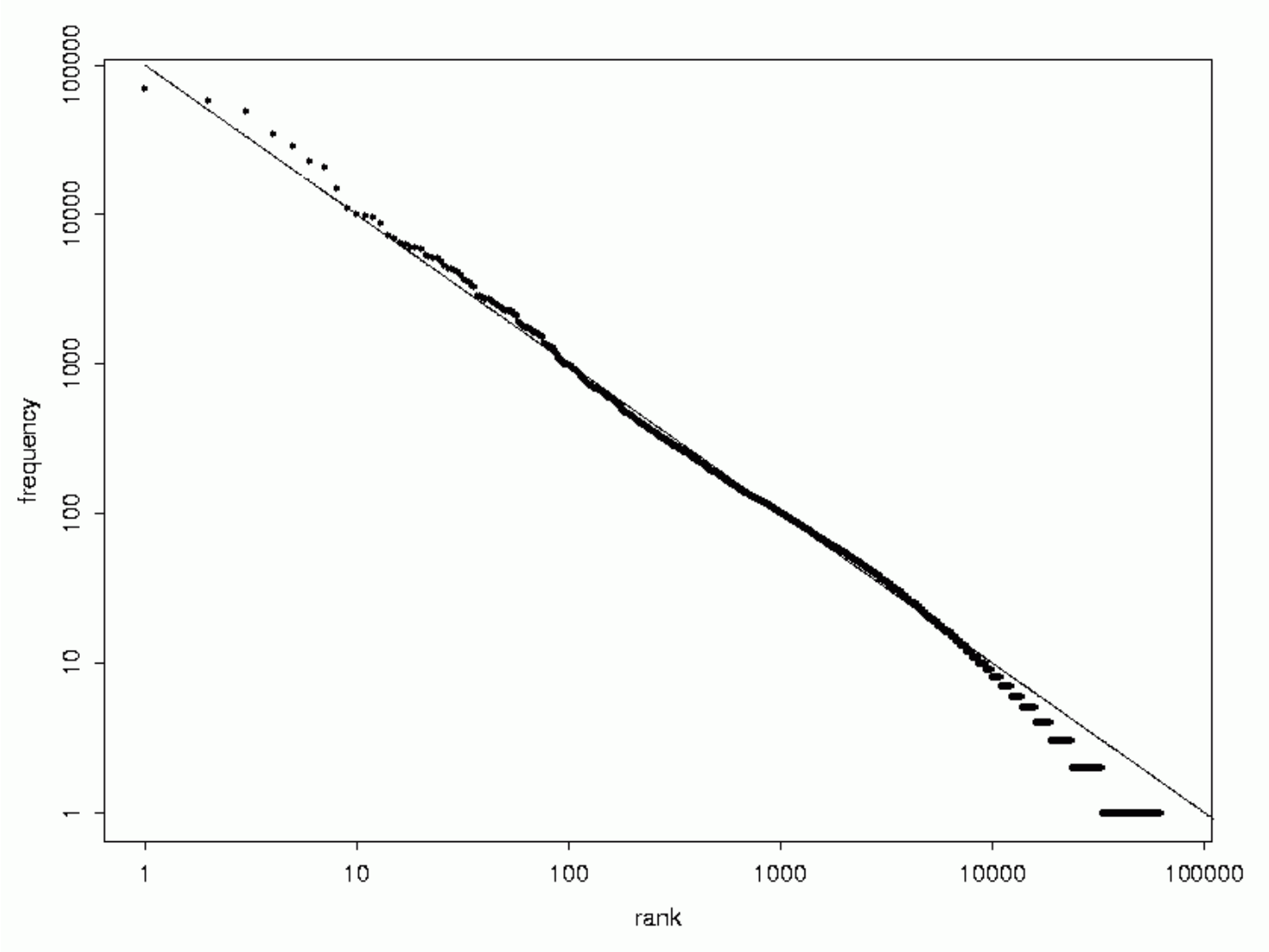
Top 50 words from 423 short TIME magazine articles

Word	Freq	r	Pr(%)	r*Pr
the	2,420,778	1	6.488	0.0649
of	1,045,733	2	2.803	0.0561
to	968,882	3	2.597	0.0779
a	892,429	4	2.392	0.0957
and	865,644	5	2.32	0.116
in	847,825	6	2.272	0.1363
said	504,593	7	1.352	0.0947
for	363,865	8	0.975	0.078
that	347,072	9	0.93	0.0837
was	293,027	10	0.785	0.0785
on	291,947	11	0.783	0.0861
he	250,919	12	0.673	0.0807
is	245,843	13	0.659	0.0857
with	223,846	14	0.6	0.084
at	210,064	15	0.563	0.0845
by	209,586	16	0.562	0.0899
it	195,621	17	0.524	0.0891
from	189,451	18	0.508	0.0914
as	181,714	19	0.487	0.0925
be	157,300	20	0.422	0.0843
were	153,913	21	0.413	0.0866
an	152,576	22	0.409	0.09
have	149,749	23	0.401	0.0923
his	142,285	24	0.381	0.0915
but	140,880	25	0.378	0.0944

Word	Freq	r	Pr(%)	r*Pr
has	136,007	26	0.365	0.0948
are	130,322	27	0.349	0.0943
not	127,493	28	0.342	0.0957
who	116,364	29	0.312	0.0904
they	111,024	30	0.298	0.0893
its	111,021	31	0.298	0.0922
had	103,943	32	0.279	0.0892
will	102,949	33	0.276	0.0911
would	99,503	34	0.267	0.0907
about	92,983	35	0.249	0.0872
i	92,005	36	0.247	0.0888
been	88,786	37	0.238	0.0881
this	87,286	38	0.234	0.0889
their	84,638	39	0.227	0.0885
new	83,449	40	0.224	0.0895
or	81,796	41	0.219	0.0899
which	80,385	42	0.215	0.0905
we	80,245	43	0.215	0.0925
more	76,388	44	0.205	0.0901
after	75,165	45	0.201	0.0907
us	72,045	46	0.193	0.0888
percent	71,956	47	0.193	0.0906
up	71,082	48	0.191	0.0915
one	70,266	49	0.188	0.0923
people	68,988	50	0.185	0.0925

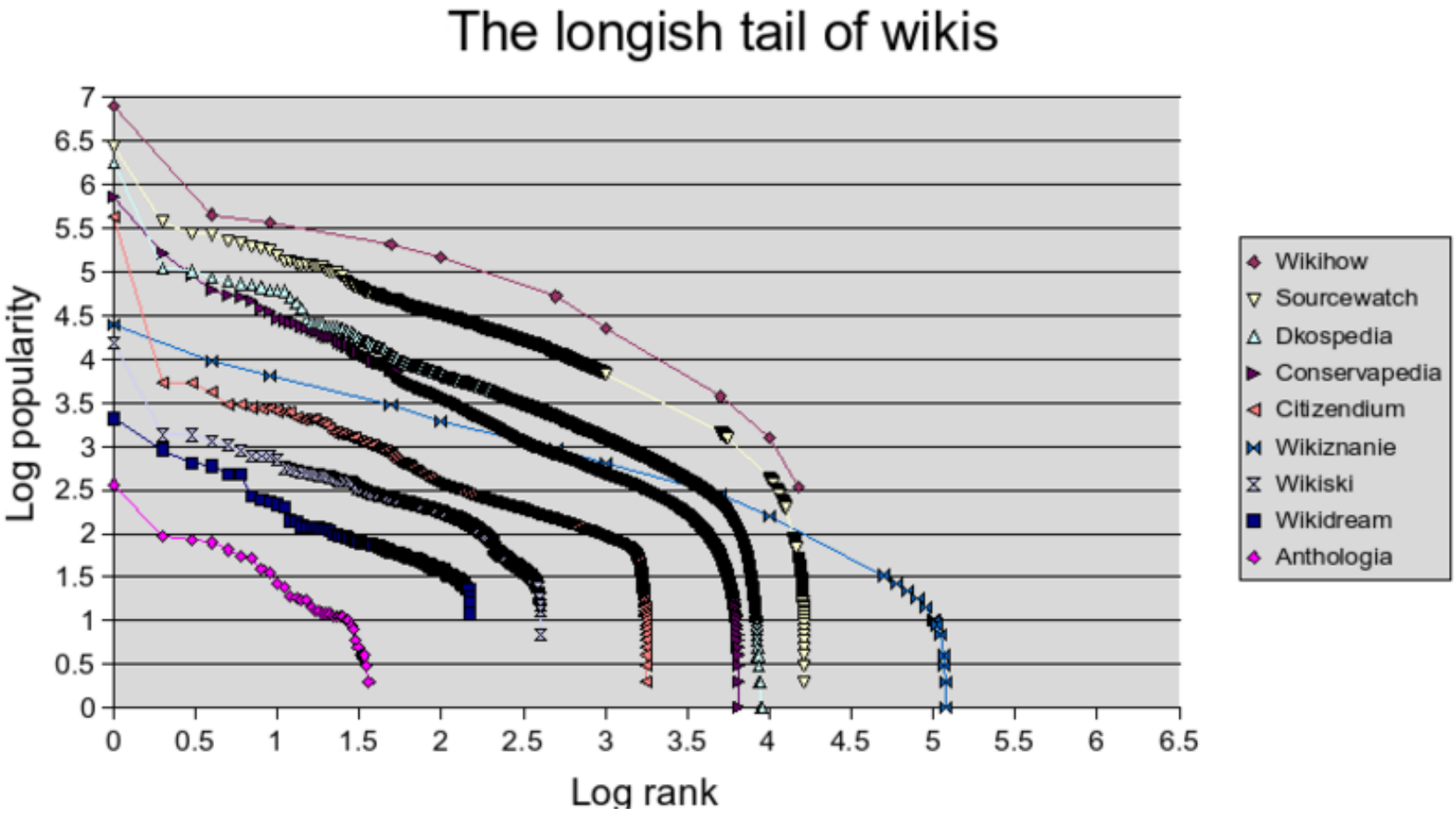
Top 50 words from 84,678 Associated Press 1989 articles

Zipf's law log-log plot: Reuters text

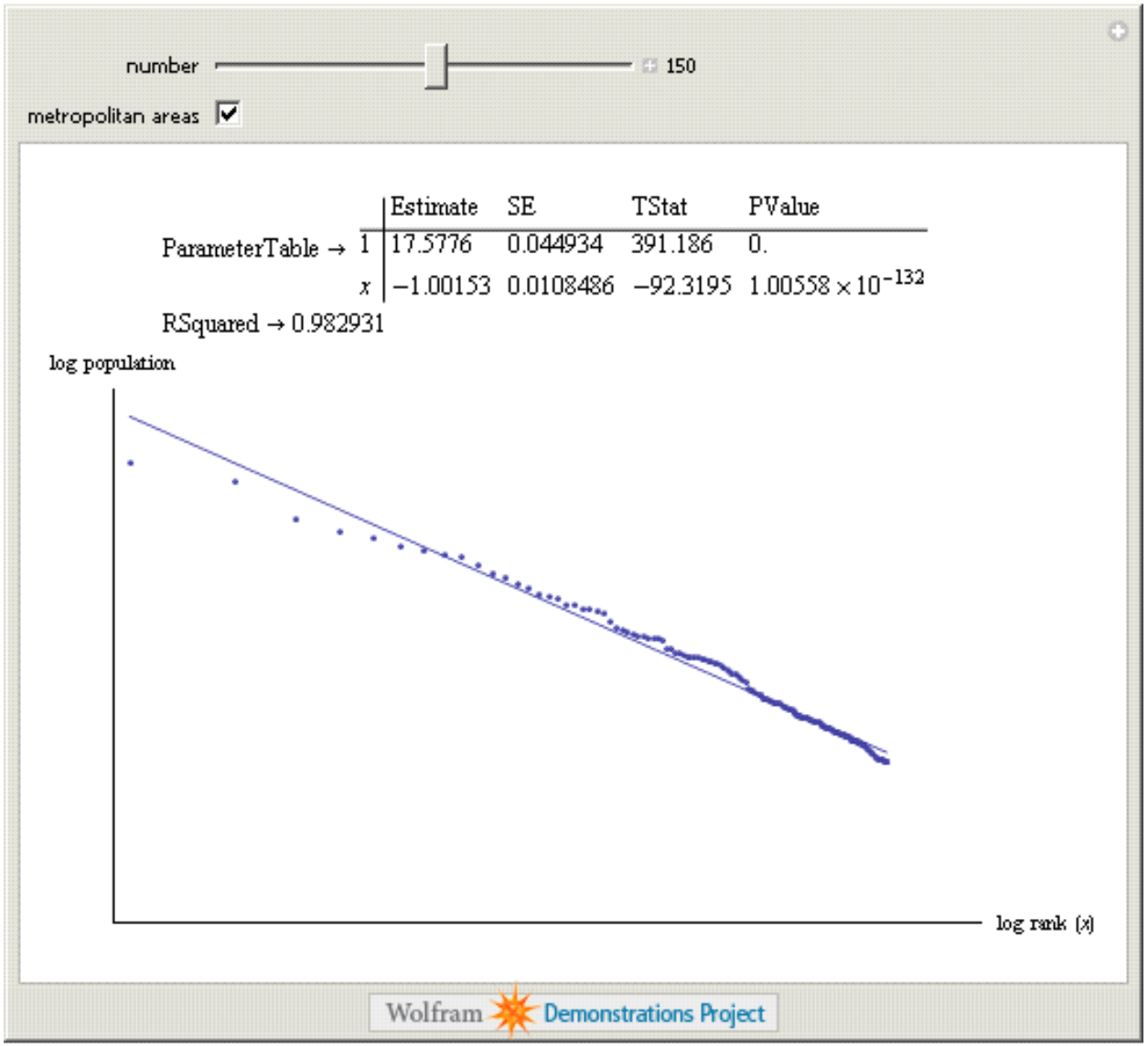


Zipf in other domains

Web page popularity



Population of Cities



Why do we observe Zipf's Law?

- Zipf's explanation was his "principle of least effort."
  - Balance between speaker's desire for a small vocabulary and hearer's desire for a large one.
- Debate (1955-61) between Mandelbrot and H. Simon over explanation.
- Li (1992) shows that just random typing of letters including a space will generate "words" with a Zipfian distribution.

Stemming / Stopwords: Impact on Index

	(distinct) terms			non-positional postings			tokens (= number of position entries in postings)		
	number	Δ%	T%	number	Δ%	T%	number	Δ%	T%
unfiltered	484,494			109,971,179			197,879,290		
no numbers	473,723	−2	−2	100,680,242	−8	−8	179,158,204	−9	−9
case folding	391,523	−17	−19	96,969,056	−3	−12	179,158,204	−0	−9
30 stop words	391,493	−0	−19	83,390,443	−14	−24	121,857,825	−31	−38
150 stop words	391,373	−0	−19	67,001,847	−30	−39	94,516,599	−47	−52
stemming	322,383	−17	−33	63,812,300	−4	−42	94,516,599	−0	−52

Scoring as the basis of ranked retrieval

Scoring as the basis of ranked retrieval

- We wish to rank documents that are more relevant higher than documents that are less relevant.
- How can we accomplish such a ranking of the documents in the collection with respect to a query?
- Assign a score to each query-document pair, say in [0, 1].
- This score measures how well document and query "match".

Query-document matching scores

- How do we compute the score of a query-document pair?
- Let's start with a one-term query.
- If the query term does not occur in the document: score should be 0.
- The more frequent the query term in the document, the higher the score
- We will look at a number of alternatives for doing this.

Take 1: Jaccard coefficient

- A commonly used measure of overlap of two sets
- Let A and B be two sets
- Jaccard coefficient:

$$\text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

$$(A \neq \emptyset \text{ or } B \neq \emptyset)$$

- JACCARD $(A, A) = 1$
- JACCARD $(A, B) = 0$ if $A \cap B = 0$
- A and B don't have to be the same size.
- Always assigns a number between 0 and 1.

Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:
  - Query: "ides of March"
  - Document "Caesar died in March"

Jaccard coefficient: Example

- What is the query-document match score that the Jaccard coefficient computes for:
  - Query: "ides of March"
  - Document "Caesar died in March"
  - JACCARD(q, d) = 1/6

What's wrong with Jaccard?

- It doesn't consider term frequency (how many occurrences a term has).
- Rare terms are more informative than frequent terms. Jaccard does not consider this information.
- We need a more sophisticated way of normalizing for the length of a document.
- Later in this lecture, we'll use $|A \cap B|/\sqrt{|A \cup B|}$ (cosine) . . .
- . . . instead of $|A \cap B|/|A \cup B|$ (Jaccard) for length normalization.

Term Frequency

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	1	1	0	0	0	1
BRUTUS	1	1	0	1	0	0
CAESAR	1	1	0	1	1	1
CALPURNIA	0	1	0	0	0	0
CLEOPATRA	1	0	0	0	0	0
MERCY	1	0	1	1	1	1
WORSER	1	0	1	1	1	0
. . .						

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth . . .
ANTHONY	157	73	0	0	0	1
BRUTUS	4	157	0	2	0	0
CAESAR	232	227	0	2	1	0
CALPURNIA	0	10	0	0	0	0
CLEOPATRA	57	0	0	0	0	0
MERCY	2	0	3	8	5	8
WORSER	2	0	1	1	1	5
. . .						

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Bag of words model

- We do not consider the order of words in a document.
- *John is quicker than Mary and Mary is quicker than John* are represented the same way.
- This is called a bag of words model.

Term frequency tf

- The term frequency $tf_{t,d}$ of term *t* in document *d* is defined as the number of times that *t* occurs in *d*.
- We want to use tf when computing query-document match scores.
- But how?
- Raw term frequency is not what we want because:
- A document with tf = 10 occurrences of the term is more relevant than a document with tf = 1 occurrence of the term.
- But not 10 times more relevant.
- Relevance does not increase proportionally with term frequency.

Instead of raw frequency: Log frequency weighting

- The log frequency weight of term t in d is defined as follows

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- $\mathrm{tf}_{t,d} \rightarrow w_{t,d}$:
  0 → 0, 1 → 1, 2 → 1.3, 10 → 2, 1000 → 4, etc.
- Score for a document-query pair: sum over terms t in both *q* and *d*:
  tf-matching-score(*q*, *d*) = $\sum_{t \in q \cap d} (1 + \log \mathrm{tf}_{t,d})$
- The score is 0 if none of the query terms is present in the document.

Exercise

- Compute the Jaccard matching score and the tf matching score for the following query-document pairs.
- q: [information on cars] d: "all you've ever wanted to know about cars"
- q: [information on cars] d: "information on trucks, information on planes, information on trains"
- q: [red cars and red trucks] d: "cops stop red cars more often"

TF-IDF Weighting

Frequency in document vs. frequency in collection

- In addition, to term frequency (the frequency of the term in the document) . . .
- . . .we also want to use the frequency of the term in the collection for weighting and ranking.

Desired weight for rare terms

- Rare terms are more informative than frequent terms.
- Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC).
- A document containing this term is very likely to be relevant.
- → We want high weights for rare terms like ARACHNOCENTRIC.

Desired weight for frequent terms

- Frequent terms are less informative than rare terms.
- Consider a term in the query that is frequent in the collection (e.g., GOOD, INCREASE, LINE).
- A document containing this term is more likely to be relevant than a document that doesn't . . .
- . . . but words like GOOD, INCREASE and LINE are not sure indicators of relevance.
- → For frequent terms like GOOD, INCREASE and LINE, we want positive weights . . .
- . . . but lower weights than for rare terms.

Document frequency

- We want high weights for rare terms like ARACHNOCENTRIC.
- We want low (positive) weights for frequent words like GOOD, INCREASE and LINE.
- We will use document frequency to factor this into computing the matching score.
- The document frequency is the number of documents in the collection that the term occurs in.

idf weight

- df_t is the document frequency, the number of documents that t occurs in.
- df_t is an inverse measure of the informativeness of term t.
- We define the idf weight of term t as follows:

$$\text{idf}_t = \log_{10} \frac{N}{\text{df}_t}$$

(N is the number of documents in the collection.)

- idf_t is a measure of the informativeness of the term.
- [log N/df_t] instead of [N/df_t] to "dampen" the effect of idf
- Note that we use the log transformation for both term frequency and document frequency.

Examples for idf

- Compute idf_t using the formula: $\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effect of idf on ranking

- idf affects the ranking of documents for queries with at least two terms.
- For example, in the query "arachnocentric line", idf weighting increases the relative weight of ARACHNOCENTRIC and decreases the relative weight of LINE.
- idf has little effect on ranking for one-term queries.

Collection frequency vs. Document frequency

word	collection frequency	document frequency
INSURANCE	10440	3997
TRY	10422	8760

- Collection frequency of *t*: number of tokens of *t* in the collection
- Document frequency of *t*: number of documents *t* occurs in
- Why these numbers?
- Which word is a better search term (and should get a higher weight)?
- This example suggests that df (and idf) is better for weighting than cf (and "icf").

tf-idf weighting

- The tf-idf weight of a term is the product of its tf weight and its idf weight.

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- tf-weight
- idf-weight
- Best known weighting scheme in information retrieval
- Note: the "-" in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf

Summary: tf-idf

- Assign a tf-idf weight for each term t in each document *d*:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- The tf-idf weight . . .
  - . . . increases with the number of occurrences within a document. (term frequency)
  - . . . increases with the rarity of the term in the collection. (inverse document frequency)

Exercise: Term, collection and document frequency

Quantity	Symbol	Definition
term frequency	$tf_{t,d}$	number of occurrences of *t* in *d*
document frequency	df_t	number of documents in the collection that *t* occurs in
collection frequency	cf_t	total number of occurrences of *t* in the collection

- Relationship between df and cf?
- Relationship between tf and cf?
- Relationship between tf and df?