Chapter IR:III

III. Text Transformation

- Text Statistics
- Parsing Documents
- Information Extraction
- Link Analysis

Text Statistics

Questions

- How many words are there?
- How often does each one occur?
- How many documents can be found for a query?
- How many documents are indexed?

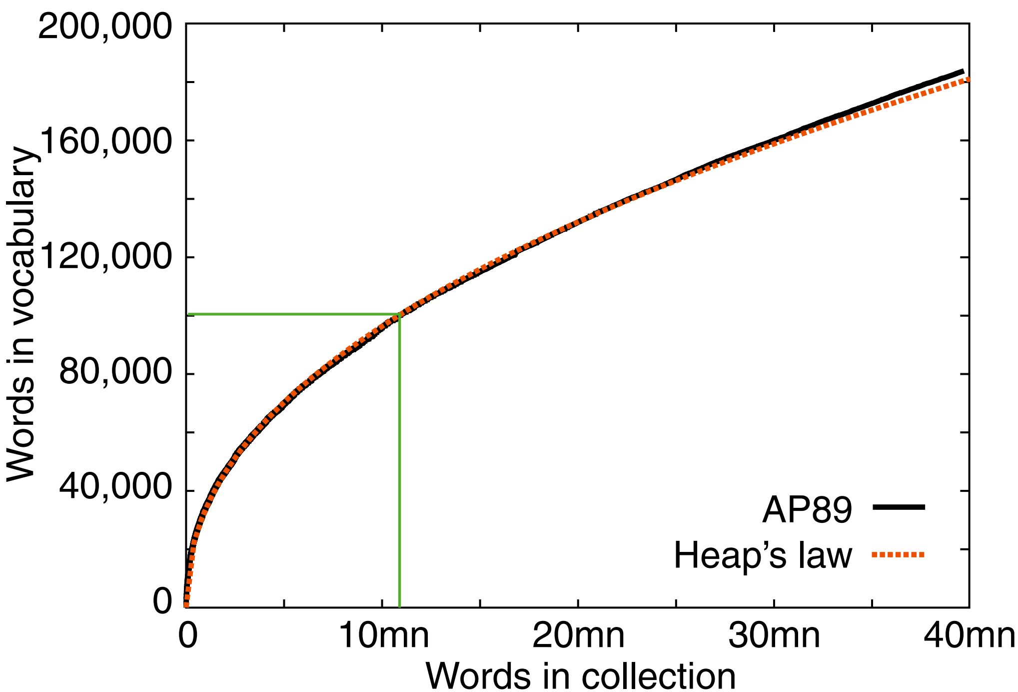
Text Statistics

Vocabulary Growth: Heaps' Law

The vocabulary V of a collection of documents grows with the collection. Vocabulary growth can be modeled with Heaps' Law:

$$|V| = k \cdot n^{\beta},$$

where n is the number of non-unique words, and k and β are collection parameters.



- Corpus: AP89
- $k = 62.95$, $\beta = 0.455$
- At $10,879,522$ words: $100,151$ predicted, $100,024$ actual.
- At $< 1,000$ words: poor predictions

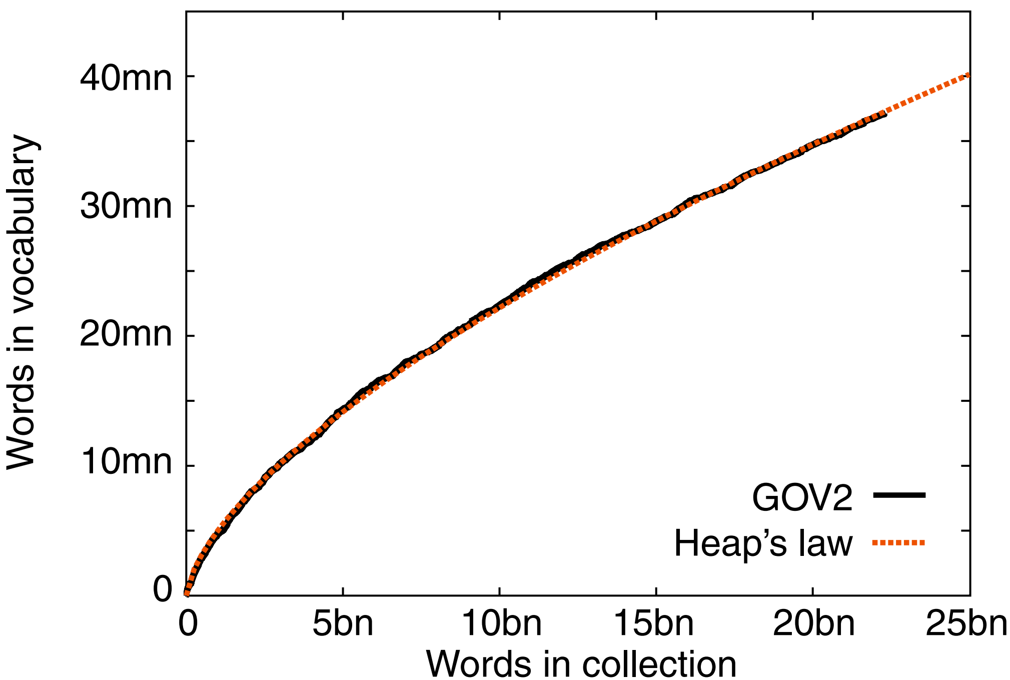
Text Statistics

Vocabulary Growth: Heaps' Law

The vocabulary V of a collection of documents grows with the collection.
Vocabulary growth can be modeled with Heaps' Law:

$$|V| = k \cdot n^{\beta},$$

where n is the number of non-unique words, and k and β are collection parameters.



- Corpus: GOV2
- $k = 7.34$, $\beta = 0.648$
- Vocabulary continuously grows in large collections
- New words include spelling errors, invented words, code, other languages, email addresses, etc.

Text Statistics

Term Frequency: Zipf's Law

The distribution of word frequencies is very *skewed*: Few words occur very frequently, many words hardly ever.

For example, the two most common English words (the, of) make up about 10% of all word occurrences in text documents. In large text samples, about 50% of the unique words occur only once.

George Kingsley Zipf, an American linguist, was among the first to study the underlying statistical relationship between the frequency of a word and its rank in terms of its frequency, formulating what is known today as Zipf's law.

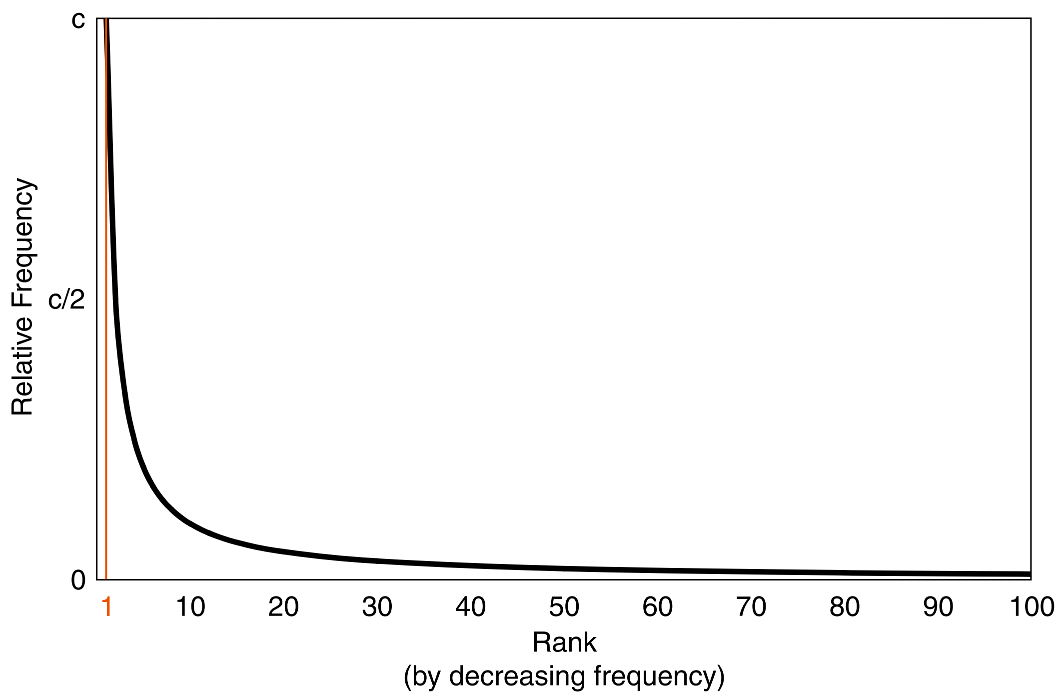
Text Statistics

Term Frequency: Zipf's Law

The relative frequency $P(w)$ of a word w in a sufficiently large text (collection) inversely correlates with its frequency rank $r(w)$ in a power law:

$$P(w) = \frac{c}{(r(w))^a},$$

where c is a constant and the exponent a is language-dependent; often $a \approx 1$.



Text Statistics

Term Frequency: Zipf's Law

A simplified formulation is $P(w) \cdot r(w) = c$.

Example: Top 50 most frequent words from AP89. Have a guess at c?

r	w	frequency	$P \cdot 100$	$P \cdot r$
1	the	2,420,778	6.09	0.061
2	of	1,045,733	2.63	0.053
3	to	968,882	2.44	0.073
4	a	892,429	2.25	0.090
5	and	865,644	2.18	0.109
6	in	847,825	2.13	0.128
7	said	504,593	1.27	0.089
8	for	363,865	0.92	0.073
9	that	347,072	0.87	0.079
10	was	293,027	0.74	0.074
11	on	291,947	0.73	0.081
12	he	250,919	0.63	0.076
13	is	245,843	0.62	0.080
14	with	223,846	0.56	0.079
15	at	210,064	0.53	0.079
16	by	209,586	0.53	0.084
17	it	195,621	0.49	0.084
18	from	189,451	0.48	0.086
19	as	181,714	0.46	0.087
20	be	157,300	0.40	0.079
21	were	153,913	0.39	0.081
22	an	152,576	0.38	0.084
23	have	149,749	0.38	0.087
24	his	142,285	0.36	0.086
25	but	140,880	0.35	0.089

r	w	frequency	$P \cdot 100$	$P \cdot r$
26	has	136,007	0.34	0.089
27	are	130,322	0.33	0.089
28	not	127,493	0.32	0.090
29	who	116,364	0.29	0.085
30	they	111,024	0.28	0.084
31	its	111,021	0.28	0.087
32	had	103,943	0.26	0.084
33	will	102,949	0.26	0.085
34	would	99,503	0.25	0.085
35	about	92,983	0.23	0.082
36	i	92,005	0.23	0.083
37	been	88,786	0.22	0.083
38	this	87,286	0.22	0.083
39	their	84,638	0.21	0.083
40	new	83,449	0.21	0.084
41	or	81,796	0.21	0.084
42	which	80,385	0.20	0.085
43	we	80,245	0.20	0.087
44	more	76,388	0.19	0.085
45	after	75,165	0.19	0.085
46	us	72,045	0.18	0.083
47	percent	71,956	0.18	0.085
48	up	71,082	0.18	0.086
49	one	70,266	0.18	0.087
50	people	68,988	0.17	0.087

Term Frequency: Zipf's Law

A simplified formulation is $P(w) \cdot r(w) = c$.

Example: Top 50 most frequent words from AP89. For English: $c \approx 0.1$.

r	w	frequency	$P \cdot 100$	$P \cdot r$
1	the	2,420,778	6.09	0.061
2	of	1,045,733	2.63	0.053
3	to	968,882	2.44	0.073
4	a	892,429	2.25	0.090
5	and	865,644	2.18	0.109
6	in	847,825	2.13	0.128
7	said	504,593	1.27	0.089
8	for	363,865	0.92	0.073
9	that	347,072	0.87	0.079
10	was	293,027	0.74	0.074
11	on	291,947	0.73	0.081
12	he	250,919	0.63	0.076
13	is	245,843	0.62	0.080
14	with	223,846	0.56	0.079
15	at	210,064	0.53	0.079
16	by	209,586	0.53	0.084
17	it	195,621	0.49	0.084
18	from	189,451	0.48	0.086
19	as	181,714	0.46	0.087
20	be	157,300	0.40	0.079
21	were	153,913	0.39	0.081
22	an	152,576	0.38	0.084
23	have	149,749	0.38	0.087
24	his	142,285	0.36	0.086
25	but	140,880	0.35	0.089

r	w	frequency	$P \cdot 100$	$P \cdot r$
26	has	136,007	0.34	0.089
27	are	130,322	0.33	0.089
28	not	127,493	0.32	0.090
29	who	116,364	0.29	0.085
30	they	111,024	0.28	0.084
31	its	111,021	0.28	0.087
32	had	103,943	0.26	0.084
33	will	102,949	0.26	0.085
34	would	99,503	0.25	0.085
35	about	92,983	0.23	0.082
36	i	92,005	0.23	0.083
37	been	88,786	0.22	0.083
38	this	87,286	0.22	0.083
39	their	84,638	0.21	0.083
40	new	83,449	0.21	0.084
41	or	81,796	0.21	0.084
42	which	80,385	0.20	0.085
43	we	80,245	0.20	0.087
44	more	76,388	0.19	0.085
45	after	75,165	0.19	0.085
46	us	72,045	0.18	0.083
47	percent	71,956	0.18	0.085
48	up	71,082	0.18	0.086
49	one	70,266	0.18	0.087
50	people	68,988	0.17	0.087

Remarks:

- Collection statistics for AP89:

Total documents	84,678
Total word occurrences	39,749,179
Vocabulary size	198,763
Words occurring > 1000 times	4,169
Words occurring once	70,064

Text Statistics

Term Frequency: Zipf's Law

For relative frequencies, c can be estimated as follows:

$$1 \;=\; \sum_{1}^{n} P(w) \;=\; \sum_{1}^{n} \frac{c}{r(w)} \;=\; c \sum_{1}^{n} \frac{1}{r(w)} \;=\; c \cdot H_n, \quad \rightsquigarrow \quad c \;=\; \frac{1}{H_n} \approx \frac{1}{\ln(n)}$$

where n is the size $|V|$ of the vocabulary V, and H_n is the n-th harmonic number.

Thus, the expected average number of occurrences of a word w in a document d of length m is

$$m \cdot P(w),$$

since $P(w)$ can be interpreted as a term occurrence probability.

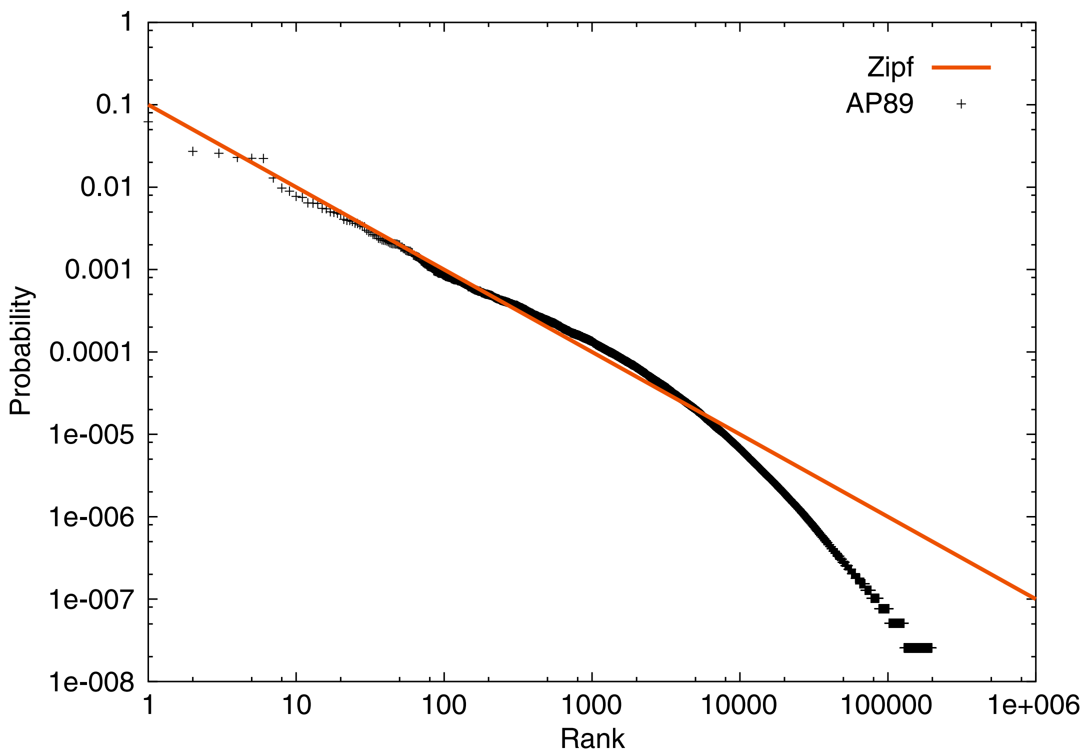
Text Statistics

Term Frequency: Zipf's Law

By logarithmization a linear form is obtained, yielding a straight line in a plot:

$$\log(P(w)) \;=\; \log(c) - a \cdot \log(r(w))$$

Example for AP89:



Remarks:

- As with all empirical laws, Zipf's law holds only approximately. While mid-range ranks of the frequency distribution fit quite well, this is less so for the lowest ranks and very high ranks (i.e., very infrequent words). The Zipf-Mandelbrot law is an extension of Zipf's law that provides for a better fit.
- Interestingly, this relation cannot only be observed for words and letters in human language texts or music score sheets, but for all kinds of natural symbol sequences (e.g., DNA). It is also true for randomly generated character sequences where one character is assigned the role of a blank space. [Li 1992]
- Independently of Zipf's law, a special case is Benford's law, which governs the frequency distribution of first digits in a number.

Text Statistics

Term Frequency: Zipf's Law

The number of words in V with a given absolute frequency x can be estimated by

$$\frac{1}{x(x+1)}$$

Derivation:

Let r_x denote the lowest rank of a word with frequency x (i.e., the one with the largest rank "number"). Then the number of words with that frequency is derived as follows:

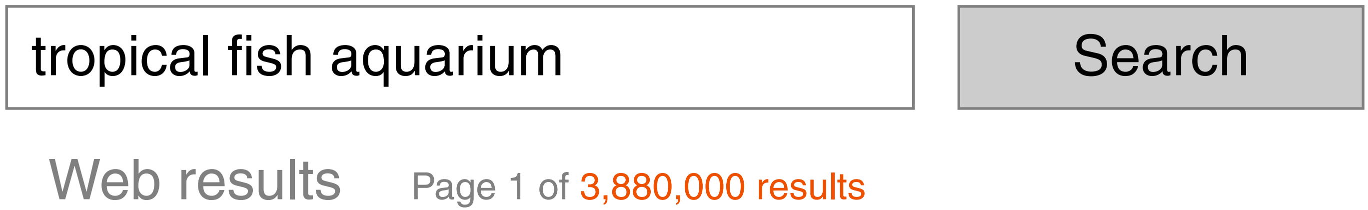
$$r_x - r_{x+1} \;=\; \frac{c_1}{p_x} - \frac{c_1}{p_{x+1}} \;=\; \frac{c_1 c_2}{x} - \frac{c_1 c_2}{x+1} \;=\; \frac{c_1 c_2}{x(x+1)},$$

where $r_x \cdot p_x = c_1$, and $p_x = x/c_2$ its relative frequency given the total number of non-unique words c_2. Its proportion of V is obtained by dividing by $|V| = r_1 = c_1 c_2$.

Observations:

- Estimations are fairly accurate for small x.
- Roughly half of all words can be expected to be unique.

Estimating Result Set Size



The total number of results is estimated, since web search engines typically do not explore the entire indexed document collection to compute the first page of results returned, but only a subset.

Approaches:

- Joint probability estimation
- Conditional probability estimation
- Initial result set-based estimation

Remarks:

- Example data from the GOV2 collection (collection size $|D|$ is 25,205,179):

Query	Document frequency
`tropical`	120,990
`fish`	1,131,855
`aquarium`	26,480
`breeding`	81,885
`tropical fish`	18,472
`tropical aquarium`	1,921
`tropical breeding`	5,510
`fish aquarium`	9,722
`fish breeding`	36,427
`aquarium breeding`	1,848
`tropical fish aquarium`	1,529
`tropical fish breeding`	3,629

Estimating Result Set Size: Joint Probability

Let $P_{df}(w)$ denote the probability of w occurring at least once in a document:

$$P_{df}(w) = \frac{df(w)}{|D|},$$

where D denotes the document collection of size $|D|$ and $df(w)$ the number of documents in D containing w, called document frequency.

The result set size of a query q of length $|q|$ words can be estimated with

$$df(q) = |D| \cdot \prod_{i=1}^{|q|} P_{df}(w_i) = \frac{\prod_{i=1}^{|q|} df(w_i)}{|D|^{|q|-1}},$$

where w_i denotes the i-th word in q. This estimation presumes word independence.

Examples:

- $df(\texttt{tropical fish aquarium}) = 5.71$ actual: 1,529 documents
- $df(\texttt{tropical fish breeding}) = 17.65$ actual: 3,629 documents

Text Statistics

Estimating Result Set Size: Conditional Probability

By exploiting word co-occurrence information, we can obtain better estimates with

$$P_{df}(q) \;=\; P_{df}(w_1 \cap w_2 \cap w_3) \;=\; P_{df}(w_1 \cap w_2) \cdot P_{df}(w_3 \mid w_1 \cap w_2),$$

where $P_{df}(w_3 \mid w_1 \cap w_2) \approx P_{df}(w_3 \mid w_x) = \max\{P_{df}(w_3 \mid w_1), P_{df}(w_3 \mid w_2)\}$ and $|q| = 3$.
Recall that $P(A \mid B) = P(A \cap B)/P(B)$. Hence

$$df(q) \;=\; |D| \cdot P_{df}(q) \;=\; \frac{df(w_1, w_2) \cdot df(w_x, w_3)}{df(w_x)}.$$

- Queries of length $|q| = 2$ need not be estimated, anymore.
- Queries of length $|q| = 3$ are typically underestimated.
- Queries of length $|q| > 3$ still require estimations based on word independence, or storing higher-order co-occurrence information.

Examples:

- $df(\texttt{tropical fish aquarium}) = 293$ actual: 1,529 documents
- $df(\texttt{tropical fish breeding}) = 841$ actual: 3,629 documents

Text Statistics

Estimating Result Set Size: Initial Result Set-based Estimation

Let $D' \subset D$ denote the documents initially scored for a query q. Then the size of the total result set in D can be estimated with

$$df(q) \;=\; |D_w| \cdot \frac{|D'_q|}{|D'|} \;=\; |D_w| \cdot \frac{|\{d \mid d \in D' \wedge q \in d\}|}{|D'|},$$

where w is the word in q with the smallest $D_w \subset D$ of documents containing w, and $D'_q \subset D'$ is the subset of initially scored documents containing all words of q.

This estimator presumes relevant documents are uniformly distributed across all documents in D_w. Why does it overestimate the result set size?

Examples:

- With $D_{\texttt{aquarium}} = 26,480$, let $|D'| = 3,000$, and $|D'_q| = 258$:
- $df(\texttt{tropical fish aquarium}) = 2,277$ actual: 1,529 documents
- With $D_{\texttt{breeding}} = 81,885$, let $|D'| = 3,000$, and $|D'_q| = 150$:
- $df(\texttt{tropical fish breeding}) = 4,094$ actual: 3,629 documents

Text Statistics

Estimating Result Set Size: Initial Result Set-based Estimation

Let $D' \subset D$ denote the documents initially scored for a query q. Then the size of the total result set in D can be estimated with

$$df(q) \;=\; |D_w| \cdot \frac{|D'_q|}{|D'|} \;=\; |D_w| \cdot \frac{|\{d \mid d \in D' \wedge q \in d\}|}{|D'|},$$

where w is the word in q with the smallest $D_w \subset D$ of documents containing w, and $D'_q \subset D'$ is the subset of initially scored documents containing all words of q.

This estimator presumes relevant documents are uniformly distributed across all documents in D_w. Overestimations result from D' containing the most “important” documents indexed. As $|D'|$ approaches $|D_w|$, estimations approach the true figure.

Examples:

- With $D_{\texttt{aquarium}} = 26,480$, let $|D'| = 6,000$, and $|D'_q| = 402$:
- $df(\texttt{tropical fish aquarium}) = 1,774$ actual: 1,529 documents

- With $D_{\texttt{breeding}} = 81,885$, let $|D'| = 6,000$, and $|D'_q| = 276$:
- $df(\texttt{tropical fish breeding}) = 3,767$ actual: 3,629 documents

Text Statistics

Estimating Collection Size: Joint Probability-based

Most search engines are black boxes to outsiders, and many do not share the size of the document collection they index, so that estimating that size has become an important task, both for academia and industry.

Given a web search engine, the size $|D|$ of the document collection D indexed can be estimated using two independently occurring words w_1 and w_2:

$$P_{df}(w_1 \cap w_2) = P_{df}(w_1) \cdot P_{df}(w_2) \quad \rightsquigarrow \quad |D| = \frac{df(w_1) \cdot df(w_2)}{df(w_1, w_2)}.$$

Averaging over many word pairs improves the estimate.

Example for GOV2:

- $df(\texttt{tropical}) = 120,990$,
  $df(\texttt{lincoln}) = 771,326$, and
  $df(\texttt{tropical, lincoln}) = 3018$.
- Then $|D| = 30,922,045$ actual: 25,205,179 documents

Text Statistics

Estimating Collection Size: Proportionality [van den Bosch 2016] [worldwidewebsize.com]

Given a web search engine, the size $|D|$ of the document collection D indexed can be estimated when presuming proportionality to a different reference collection D':

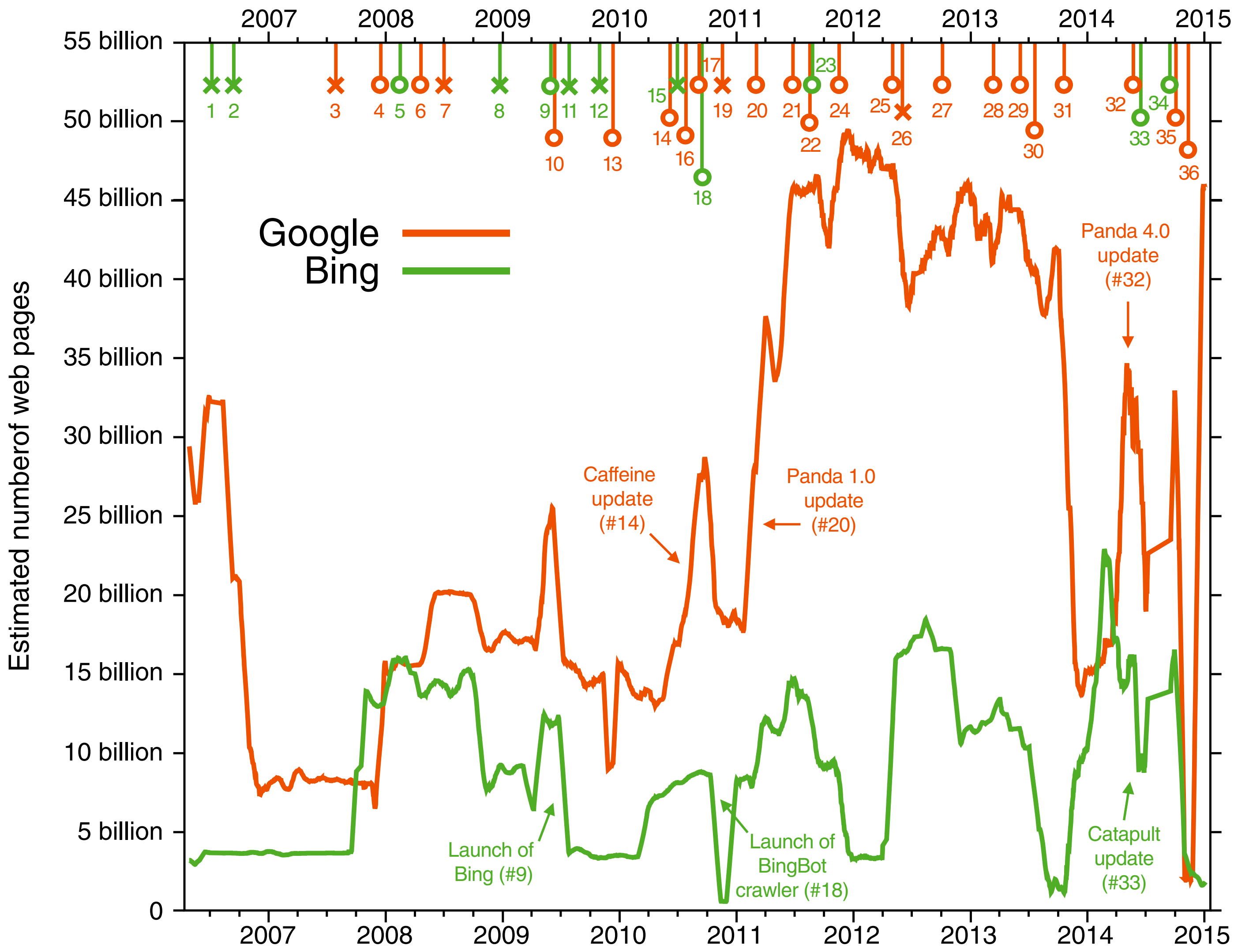
$$P_{df}^{(D)}(w) = P_{df}^{(D')}(w) \quad \rightsquigarrow \quad |D| = \frac{df_D(w) \cdot |D'|}{df_{D'}(w)},$$

where df_D computes the document frequency for D.

Averaging over words of varying frequencies improves the estimate.

Text Statistics

Estimating Collection Size: Proportionality [van den Bosch 2016] [worldwidewebsize.com]



Remarks:

1. [2006-07-04] MSN Search outage
2. [2006-09-11] Launch of (improvements to) Live Search
3. [2007-07-31] Update to supplemental results indexing
4. [2007-12-18] No more supplemental index; whole index is searched for every query
5. [2008-02-12] Crawler improvements for Live Search
6. [2008-04-11] Improved crawling of HTML forms
7. [2008-06-30] Improved Flash indexing
8. [2008-12-11] First experiments with MSNBot 2.0
9. [2009-05-28] Launch of Bing
10. [2009-06-18] Improved Flash indexing
11. [2009-07-31] Bing and Yahoo! team up on search
12. [2009-11-04] MSNBot 2.0
13. [2009-12-07] Updates to real-time search
14. [2010-06-08] Launch of new web indexing system Caffeine
15. [2010-06-28] Experiments with BingBot crawler
16. [2010-07-29] Improved Flash & AJAX indexing
17. [2010-08-31] Google indexes SVG
18. [2010-09-03] Launch of BingBot crawler
19. [2010-11-11] Improved Flash indexing
20. [2011-02-24] Panda Refresh (update to promote (English) high-quality sites more)
21. [2011-06-21] Panda 2.2

22. [2011-08-12] Panda (rolled out to all languages)
23. [2011-08-15] Gradual roll-out of Tiger indexing architecture
24. [2011-11-03] Panda (update, affects 35% of queries)
25. [2012-04-24] Penguin update (targeting Web spam, impacting around 3.1% of queries)
26. [2012-05-26] Penguin 2 update (impacting less than 0.1% of queries)
27. [2012-10-05] Penguin 3 update (impacting around 0.3% of queries)
28. [2013-03-12] Panda update
29. [2013-05-22] Penguin 4 (v2.0, impacting 2.3% of queries)
30. [2013-07-18] Panda update
31. [2013-10-04] Penguin 5 (v2.1, impacting around 1% of queries)
32. [2014-05-21] Panda 4.0
33. [2014-06-18] Launch of Bing Catapult
34. [2014-09-09] Improved spam filtering
35. [2014-09-26] Panda 4.1 (3-5% of queries affected)
36. [2014-10-17] Penguin 6 (v3.0, impacting less than 1% English queries)