INFO 6010
Computional Methods
for
Information Science Research

Meeting 3: Big Data and Power Laws

Paul Ginsparg

Cornell University, Ithaca, NY

8 Feb 2013

More Statistical Methods

Peter Norvig, "How to Write a Spelling Corrector"
`http://norvig.com/spell-correct.html`

(See video:
`http://www.youtube.com/watch?v=yvDCzhbjYWs`
"The Unreasonable Effectiveness of Data", given 23 Sep 2010.)

Additional related references:
`http://doi.ieeecomputersociety.org/10.1109/MIS.2009.36`
A. Halevy, P. Norvig, F. Pereira,
The Unreasonable Effectiveness of Data,
Intelligent Systems Mar/Apr 2009 (copy at resources/unrealdata.pdf)

`http://norvig.com/ngrams/ch14.pdf`
P. Norvig, "Natural Language Corpus Data"

A little theory

Find the correction c that maximizes the probability of c given the original word w:

$$\operatorname{argmax}_c \; P(c|w)$$

By Bayes' Theorem, equivalent to $\operatorname{argmax}_c \; P(w|c)P(c)/P(w)$. $P(w)$ the same for every possible c, so ignore, and consider:

$$\operatorname{argmax}_c \; P(w|c)P(c) \; .$$

Three parts :

- $P(c)$, the probability that a proposed correction c stands on its own. The language model: "how likely is c to appear in an English text?" ($P($"the"$)$ high, $P($"zxzxzxzyyy"$)$ near zero)
- $P(w|c)$, the probability that w would be typed when author meant c. The error model: "how likely is author to type w by mistake instead of c?"
- argmax_c, the control mechanism: choose c that gives the best combined probability score.

Example

$w =$ "thew"

- two candidate corrections $c =$ "the" and $c =$ "thaw".
- which has higher $P(c|w)$?
- "thaw" has only small change "a" to "e"
- "the" is a very common word, and perhaps the typist's finger slipped off the "e" onto the "w".

To estimate $P(c|w)$, have to consider both the probability of c and the probability of the change from c to w

[Recall the joint probability "p of A given B", written $P(A|B)$, for events A and B, can be estimated by counting the number of times that A and B both occur, and dividing by the total number of times B occurs. Intuitively it is the fraction of times A occurs out of the total times that B occurs.]

Complete Spelling Corrector

```
import re, collections

def words(text): return re.findall('[a-z]+', text.lower())

def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model

NWORDS = train(words(file('big.txt').read()))

alphabet = 'abcdefghijklmnopqrstuvwxyz'
```

```
def edits1(word):
    s = [(word[:i], word[i:]) for i in range(len(word) + 1)]
    deletes = [a + b[1:] for a, b in s if b]
    transposes = [a + b[1] + b[0] + b[2:] for a, b in s if len(b)>1]
    replaces = [a + c + b[1:] for a, b in s for c in alphabet if b]
    inserts = [a + c + b for a, b in s for c in alphabet]
    return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

def known(words): return set(w for w in words if w in NWORDS)

def correct(word):
    candidates = known([word]) or known(edits1(word))
                                or known_edits2(word) or [word]
    return max(candidates, key=NWORDS.get)
```

(For word of length n: n deletions, n-1 transpositions, 26n alterations, and 26(n+1) insertions, for a total of 54n+25 at edit distance 1)

Improvements

language model $P(c)$: need more words. add -ed to verb or -s to noun, -ly for adverbs

bad probabilities: wrong word appears more frequently?(didn't happen)

error model $P(w|c)$: sometimes edit distance 2 is better
('adres' to 'address', not 'acres')
or wrong word of many at edit distance 1
(in addition better error model permits adding more obscure words)
allow edit distance 3?

best improvement:
look for context ('they where going', 'There's no there thear')
$\Rightarrow$ Use n-grams
(See Whitelaw et al. (2009), "Using the Web for Language Independent Spellchecking and Autocorrection": Precision, recall, F1, classification accuracy)

Outline

More Data

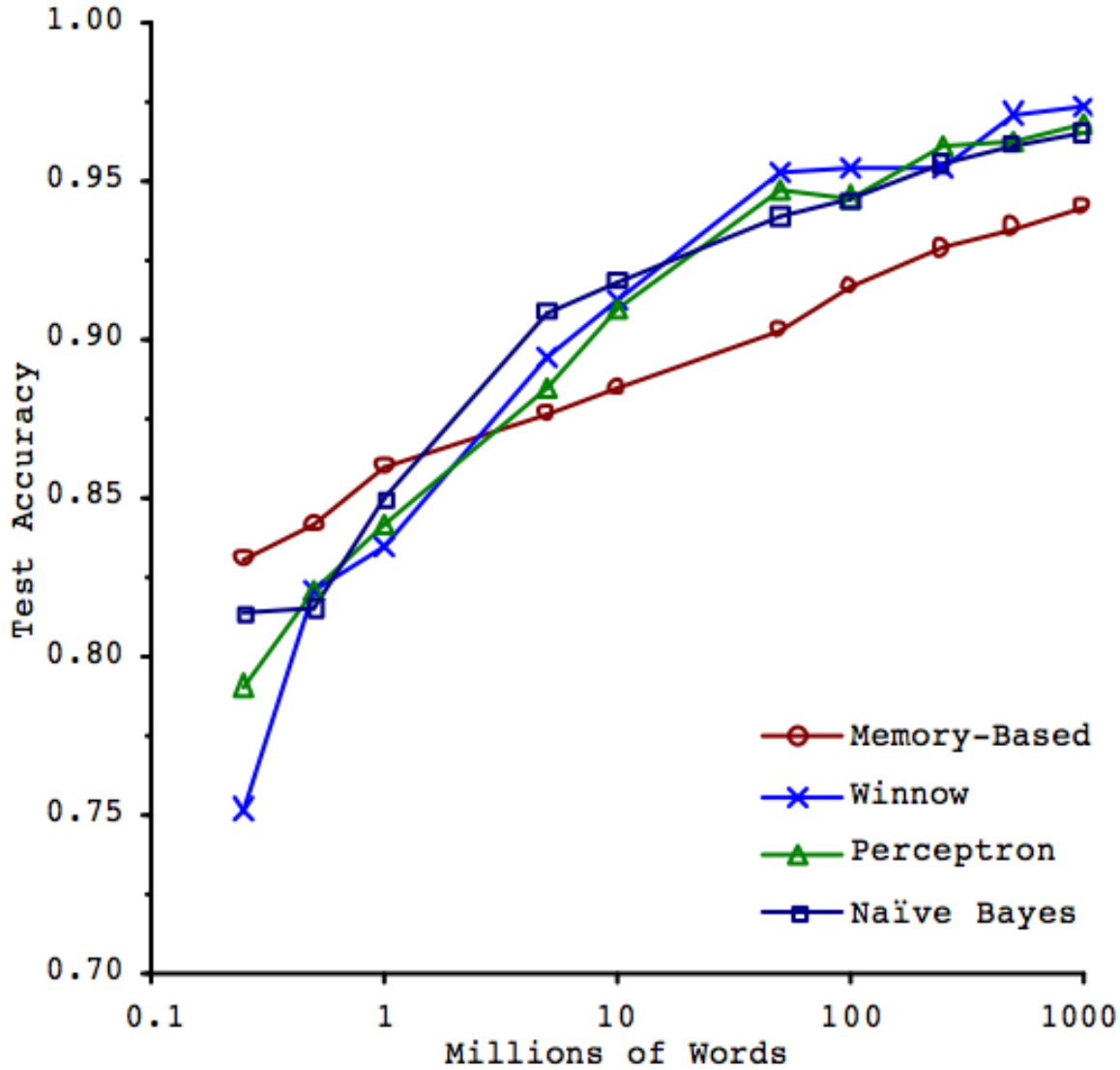


Figure 1. Learning Curves for Confusion Set Disambiguation

http://acl.ldc.upenn.edu/P/P01/P01-1005.pdf

Scaling to Very Very Large Corpora for Natural Language Disambiguation

M. Banko and E. Brill (2001)

More Data for this Task

http://acl.ldc.upenn.edu/P/P01/P01-1005.pdf
Scaling to Very Very Large Corpora for Natural Language Disambiguation
M. Banko and E. Brill (2001)

The amount of readily available on-line text has reached hundreds of billions of words and continues to grow. Yet for most core natural language tasks, algorithms continue to be optimized, tested and compared after training on corpora consisting of only one million words or less. In this paper, we evaluate the performance of different learning methods on a prototypical natural language disambiguation task, confusion set disambiguation, when trained on orders of magnitude more labeled data than has previously been used. We are fortunate that for this particular application, correctly labeled training data is free. Since this will often not be the case, we examine methods for effectively exploiting very large corpora when labeled data comes at a cost.

(Confusion set disambiguation is the problem of choosing the correct use of a word, given a set of words with which it is commonly confused. Example confusion sets include: {principle , principal}, {then , than}, {to , two , too} , and {weather,whether}.)

Segmentation

- nowisthetimeforallgoodmentocometothe
- Probability of a segmentation = P(first word) $\times$ P(rest)
- Best segmentation = one with highest probability
- P(word) = estimated by counting

Trained on 1.7B words English, 98% word accuracy

Spelling with Statistical Learning

- Probability of a spelling correction, c = P(c as a word) × P(original is a typo for c)
- Best correction = one with highest probability
- P(c as a word) = estimated by counting
- P(original is a typo for c) = proportional to number of changes

Similarly for speech recognition, using language model $p(c)$ and acoustic model $p(s|c)$
(Russel & Norvig, "Artificial Intelligence", section 24.7)

And others

- Statistical Machine Translation
  - Collect parallel texts ("Rosetta stones"), Align (Brants, Popat, Xu, Och, Dean (2007), "Large Language Models in Machine Translation")
- fill in occluded portions of photos (Hayes and Efros, 2007)

General "Big Data" Procedure

- Define a probabilistic model
  (i.e., use data to create language model, a probability distribution ver all strings in the language, learned from corpus, and use model to determine probability of candidates)
- Enumerate candidates
  (e.g., segmentations, corrected spellings)
- Choose the most probable candidate:

$$\text{best} = \text{argmax}_{c \in \text{candidates}} P(c)$$

Python: best = max(candidates, key=P)

Big Data = Simple Algorithm

back to segmentation

e.g., unigram model for segmentation:

$$P(w_1 \dots w_n) = P(w_1) \dots P(w_n)$$

To segment 'wheninrome', consider candidates such as "when in rome", and compute P(when) $\times$ P(in) $\times$ P(rome).
Gives best answer If product is larger than any other candidate's.

'wheninthecourseofhumaneventsitbecomesnecessary' has 35 trillion segmentations, but can be read by finding probable words in sequence (not by considering all 2^{n-1} segmentations)

So use the largest product recursively: P(first) $\times$ P(remaining)

Other Tasks

- Secret codes
- Language Identification
- Spam Detection and Other Classification Tasks
- Author Identification (Stylometry)

Statistical Machine Translation

Google *n*-gram corpus created by researchers in the machine translation group (released 2006).
Translating from foreign language (f) into English (e) similar to correcting misspelled words.
The best English translation is modeled as:

$$\text{best} = \text{argmax}_e P(e|f) = \text{argmax}_e P(f|e)P(e)$$

where $P(e)$ is the language model for English, which is estimated by the word *n*-gram data, and $P(f|e)$ is the translation model, learned from a bilingual corpus (where pairs of documents are marked as translations of each other). Although top systems make use of many linguistic features, including parts of speech and syntactic parses of the sentences, seems that majority of knowledge necessary for translation resides in the *n*-gram data.

Outline

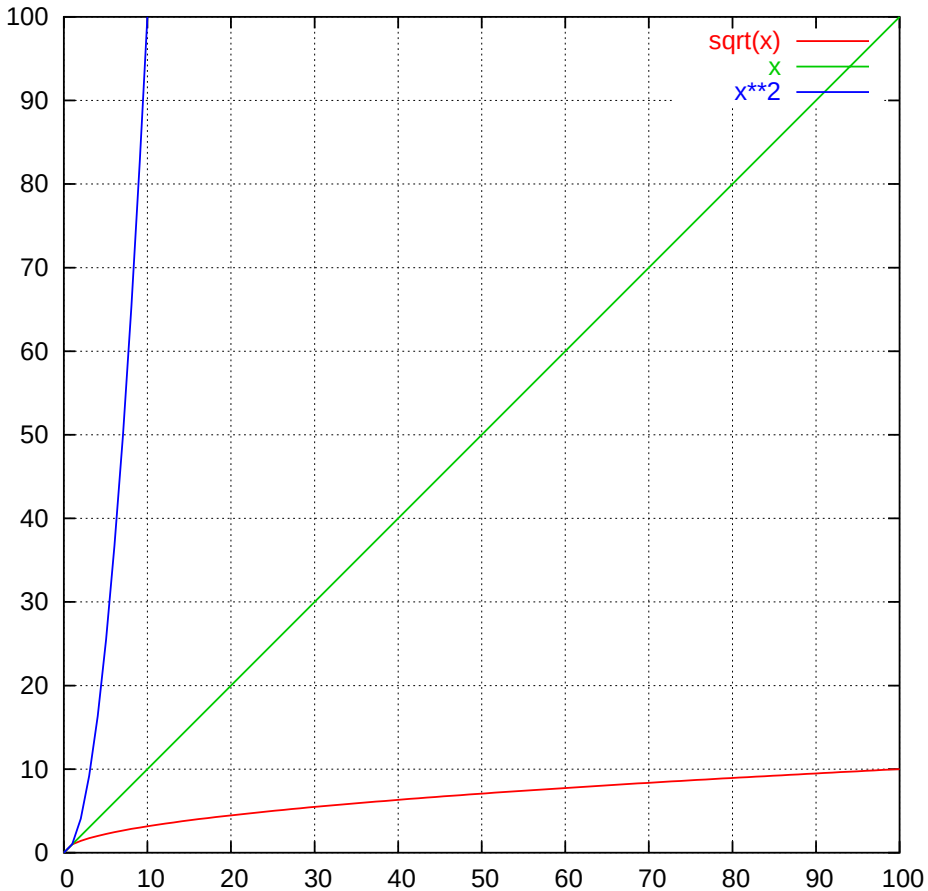
How big is the term vocabulary?

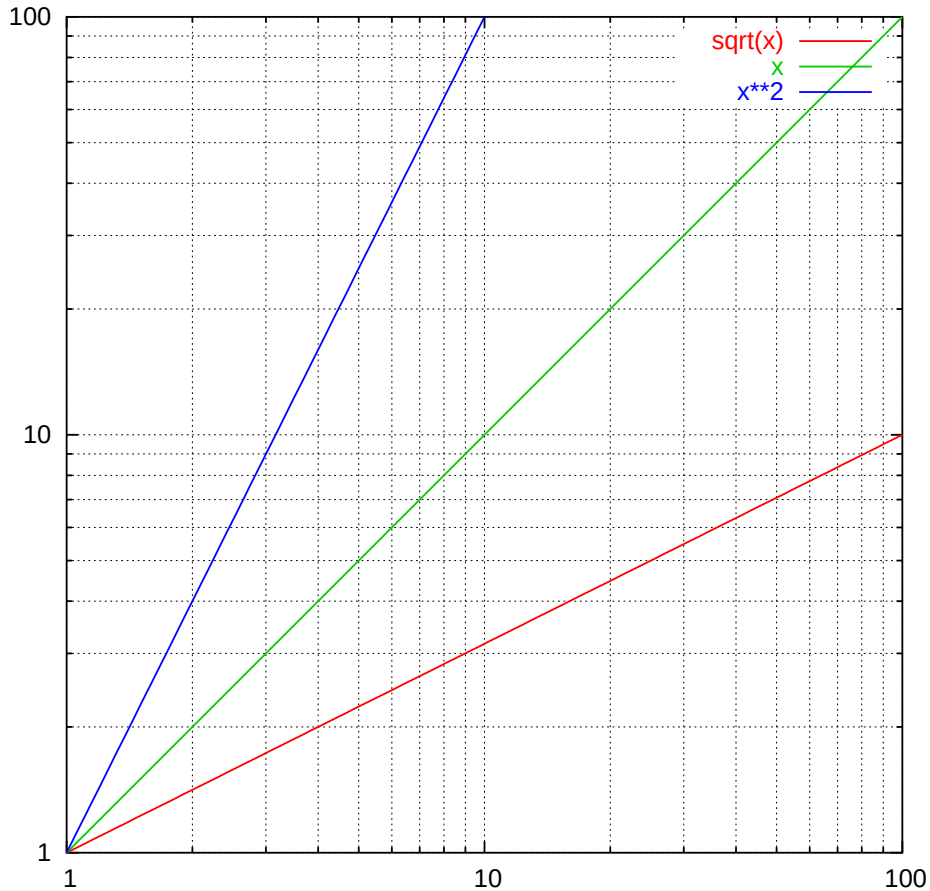
- That is, how many distinct words are there?
- Can we assume there is an upper bound?
- Not really: At least $70^{20} \approx 10^{37}$ different words of length 20.
- The vocabulary will keep growing with collection size.
- Heaps' law: $M = kT^b$
- M is the size of the vocabulary, T is the number of tokens in the collection.
- Typical values for the parameters k and b are: $30 \leq k \leq 100$ and $b \approx 0.5$.
- Heaps' law is linear in log-log space.
  - It is the simplest possible relationship between collection size and vocabulary size in log-log space.
  - Empirical law

Power Laws in log-log space

$y = cx^k$ (k=1/2,1,2)

$\log_{10} y = k * \log_{10} x + \log_{10} c$



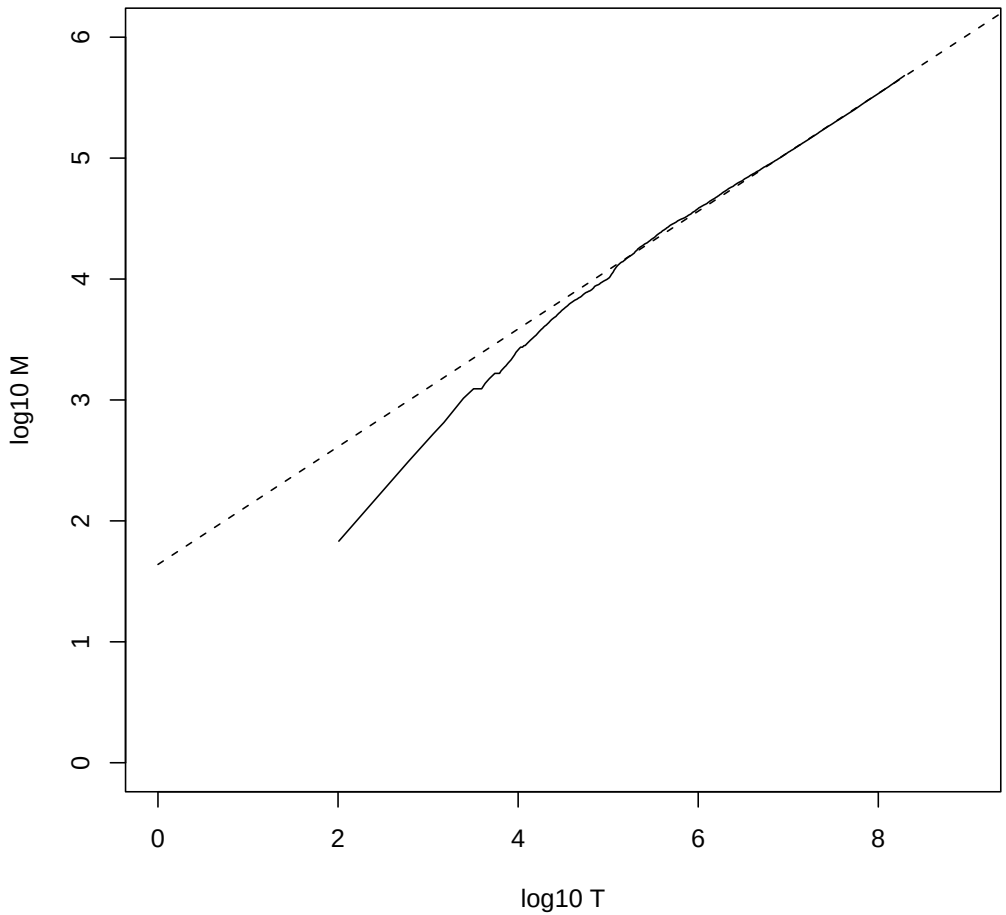


Model collection: The Reuters collection

symbol	statistic	value
N	documents	800,000
L	avg. # word tokens per document	200
M	word types	400,000
	avg. # bytes per word token (incl. spaces/punct.)	6
	avg. # bytes per word token (without spaces/punct.)	4.5
	avg. # bytes per word type	7.5
T	non-positional postings	100,000,000

1Gb of text sent over Reuters newswire 20 Aug '96 – 19 Aug '97

Heaps' law for Reuters



Vocabulary size M as a function of collection size T (number of tokens) for Reuters-RCV1. For these data, the dashed line $\log_{10} M = 0.49 * \log_{10} T + 1.64$ is the best least squares fit.
Thus, $M = 10^{1.64} T^{0.49}$
and
$k = 10^{1.64} \approx 44$
and
$b = 0.49$.

$M = kT^b = 44T^{.49}$

Empirical fit for Reuters

- Good, as we just saw in the graph.
- Example: for the first 1,000,020 tokens Heaps' law predicts 38,323 terms:

$$44 \times 1{,}000{,}020^{0.49} \approx 38{,}323$$

- The actual number is 38,365 terms, very close to the prediction.
- Empirical observation: fit is good in general.

Zipf's law

- Now we have characterized the growth of the vocabulary in collections.
- We also want to know how many frequent vs. infrequent terms we should expect in a collection.
- In natural language, there are a few very frequent terms and very many very rare terms.
- Zipf's law (linguist/philologist George Zipf, 1935):
  The i^{th} most frequent term has frequency proportional to $1/i$.
- $\text{cf}_i \propto \frac{1}{i}$
- cf_i is collection frequency: the number of occurrences of the term t_i in the collection.

http://en.wikipedia.org/wiki/Zipf's_law

Zipf's law: the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, which occurs twice as often as the fourth most frequent word, etc. Brown Corpus:

- "the": 7% of all word occurrences (69,971 of $\gtrsim$1M).
- "of": $\sim$3.5% of words (36,411)
- "and": 2.9% (28,852)

Only 135 vocabulary items account for half the Brown Corpus.

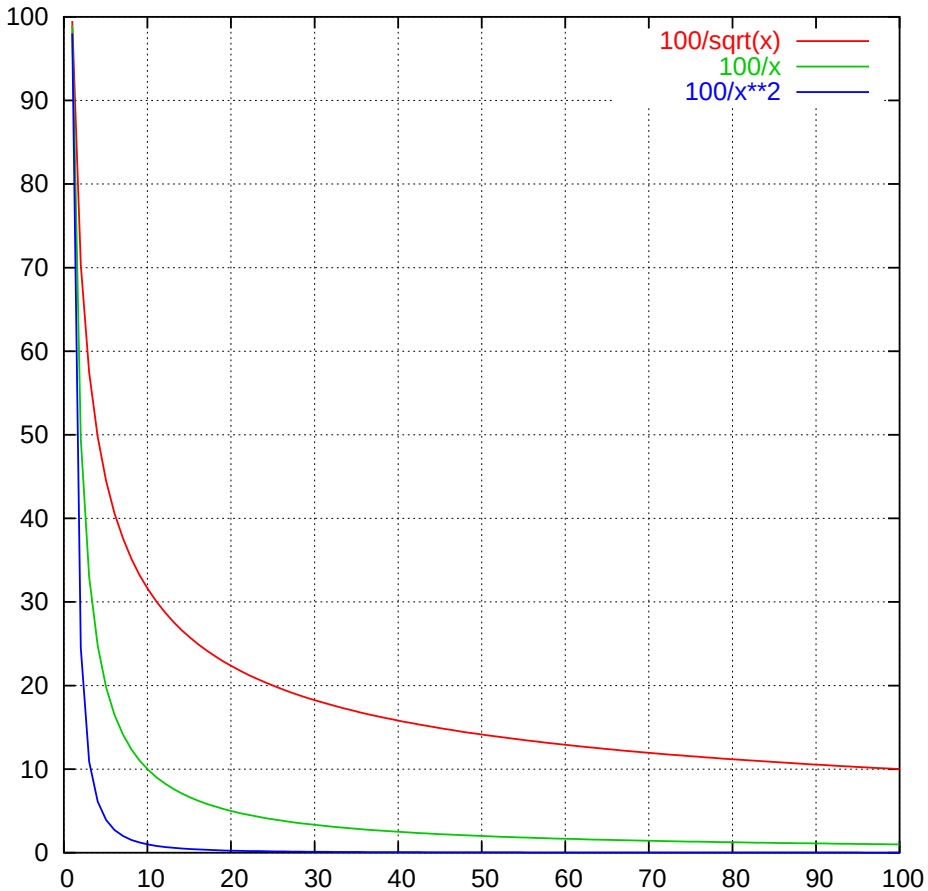
The Brown University Standard Corpus of Present-Day American English is a carefully compiled selection of current American English, totaling about a million words drawn from a wide variety of sources . . . for many years among the most-cited resources in the field.

Zipf's law

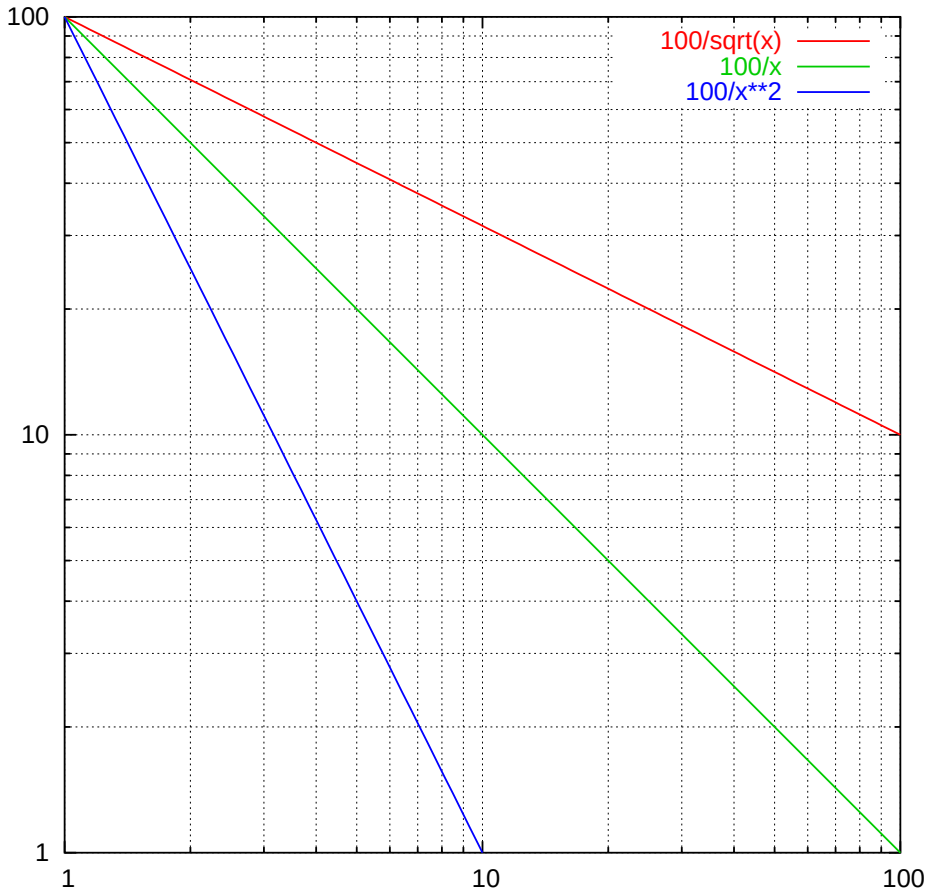
- Zipf's law: The i^{th} most frequent term has frequency proportional to $1/i$.
- $\text{cf}_i \propto \frac{1}{i}$
- cf is collection frequency: the number of occurrences of the term in the collection.
- So if the most frequent term (*the*) occurs cf_1 times, then the second most frequent term (*of*) has half as many occurrences $\text{cf}_2 = \frac{1}{2}\text{cf}_1$ ...
- ... and the third most frequent term (*and*) has a third as many occurrences $\text{cf}_3 = \frac{1}{3}\text{cf}_1$ etc.
- Equivalent: $\text{cf}_i = ci^k$ and $\log \text{cf}_i = \log c + k \log i$ (for $k = -1$)
- Example of a power law

Power Laws in log-log space

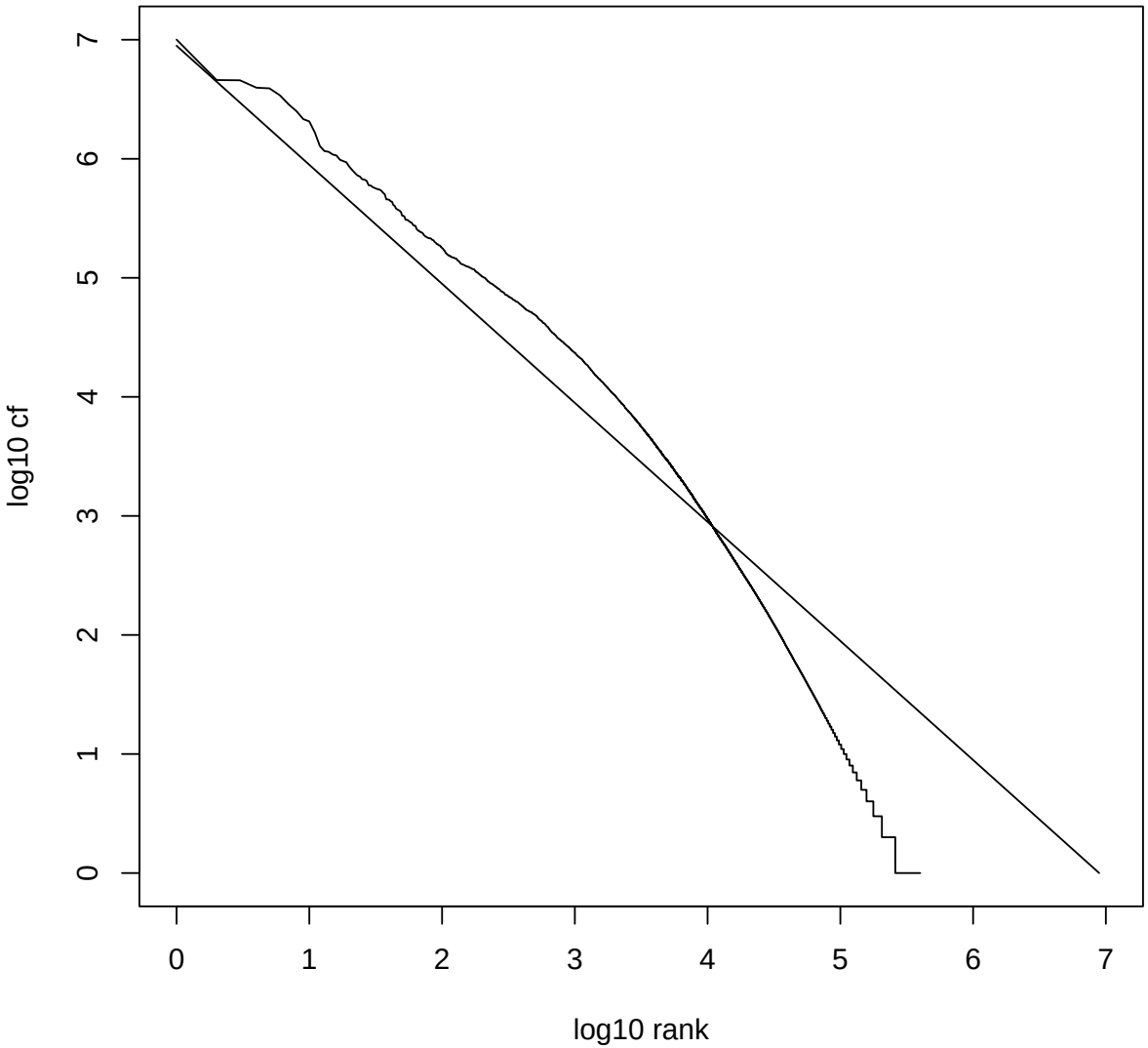
$y = cx^{-k}$ (k=1/2,1,2)



$\log_{10} y = -k * \log_{10} x + \log_{10} c$



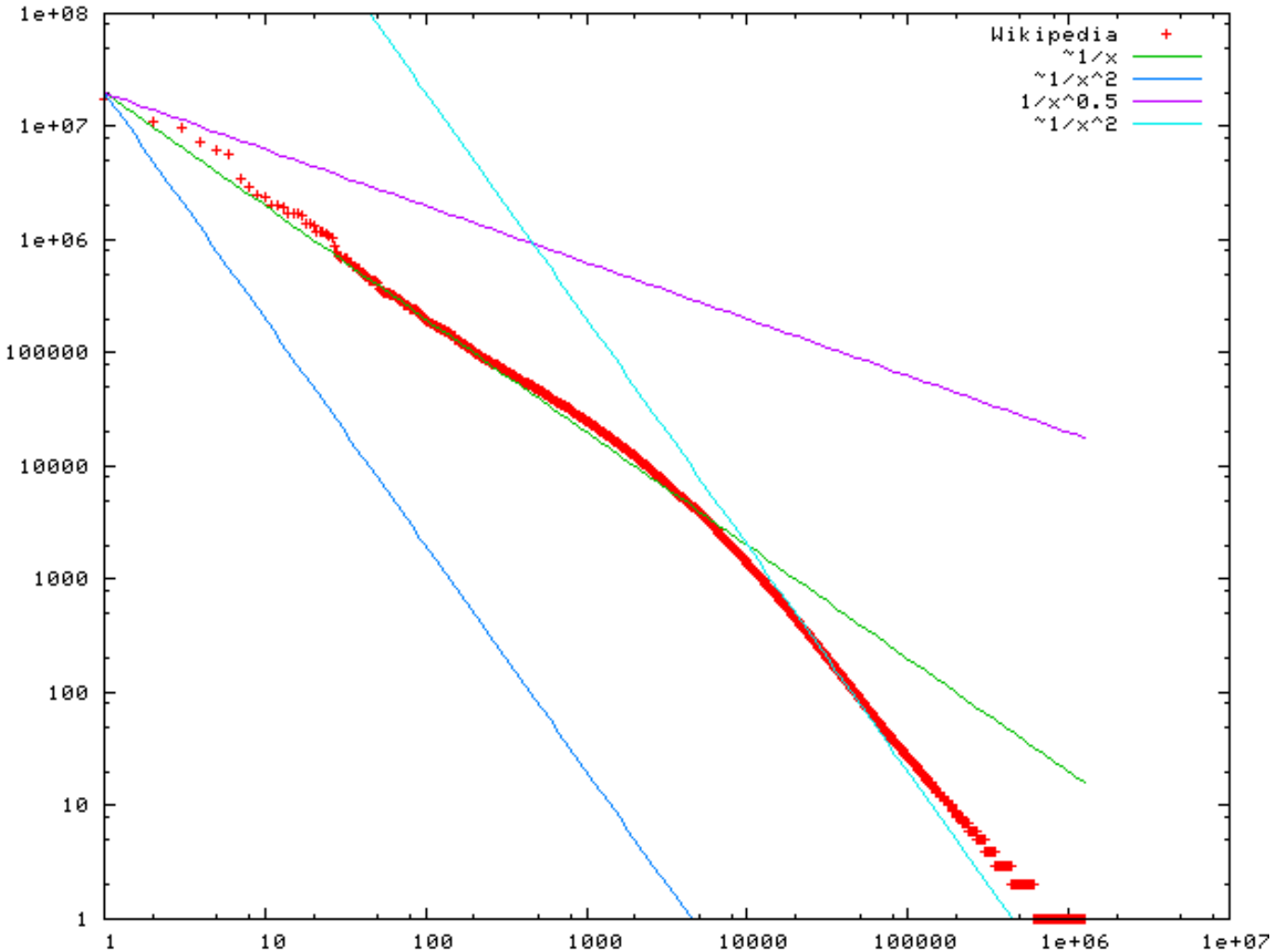
Zipf's law for Reuters



Fit far from perfect, but nonetheless key insight:

Few frequent terms, many rare terms.

more from http://en.wikipedia.org/wiki/Zipf's_law



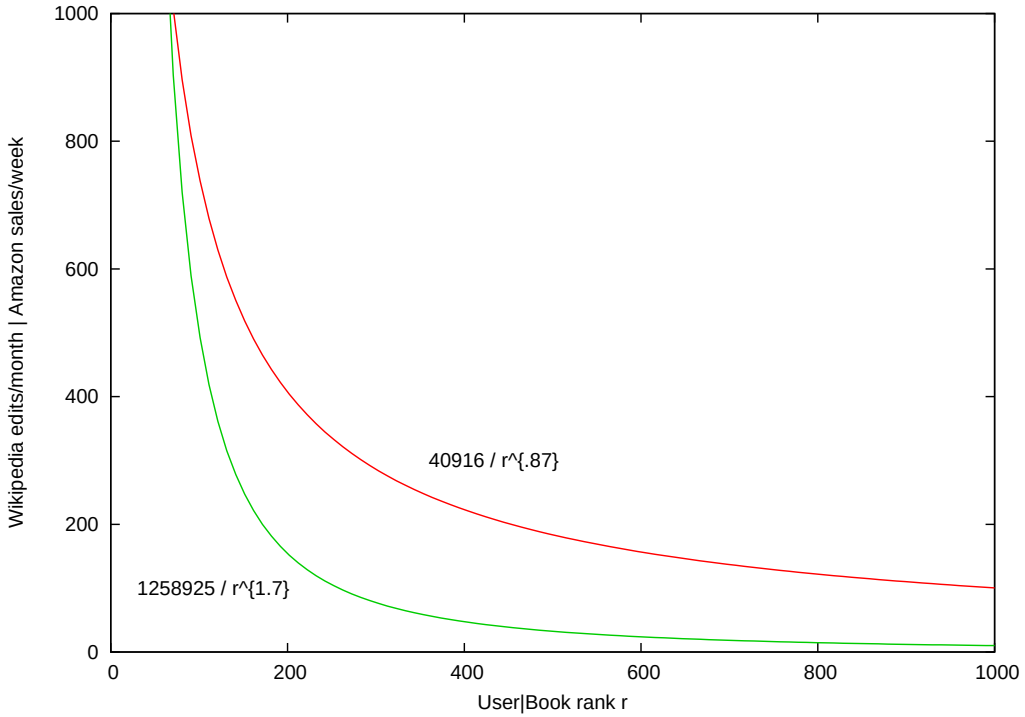
"A plot of word frequency in Wikipedia (27 Nov 2006). The plot is in log-log coordinates. x is rank of a word in the frequency table; y is the total number of the words occurrences. Most popular words are "the", "of" and "and", as expected. Zipf's law corresponds to the upper linear portion of the curve, roughly following the green (1/x) line."

Power laws more generally

E.g., consider power law distributions of the form $c\, r^{-k}$, describing the number of book sales versus sales-rank r of a book, or the number of Wikipedia edits made by the r^{th} most frequent contributor to Wikipedia.

- Amazon book sales: $c\, r^{-k}$, $k \approx .87$
- number of Wikipedia edits: $c\, r^{-k}$, $k \approx 1.7$

(More on power laws and the long tail here:
Networks, Crowds, and Markets:
Reasoning About a Highly Connected World
by David Easley and Jon Kleinberg
Chpt 18: `http://www.cs.cornell.edu/home/kleinber/networks-book/networks-book-ch18.pdf`)

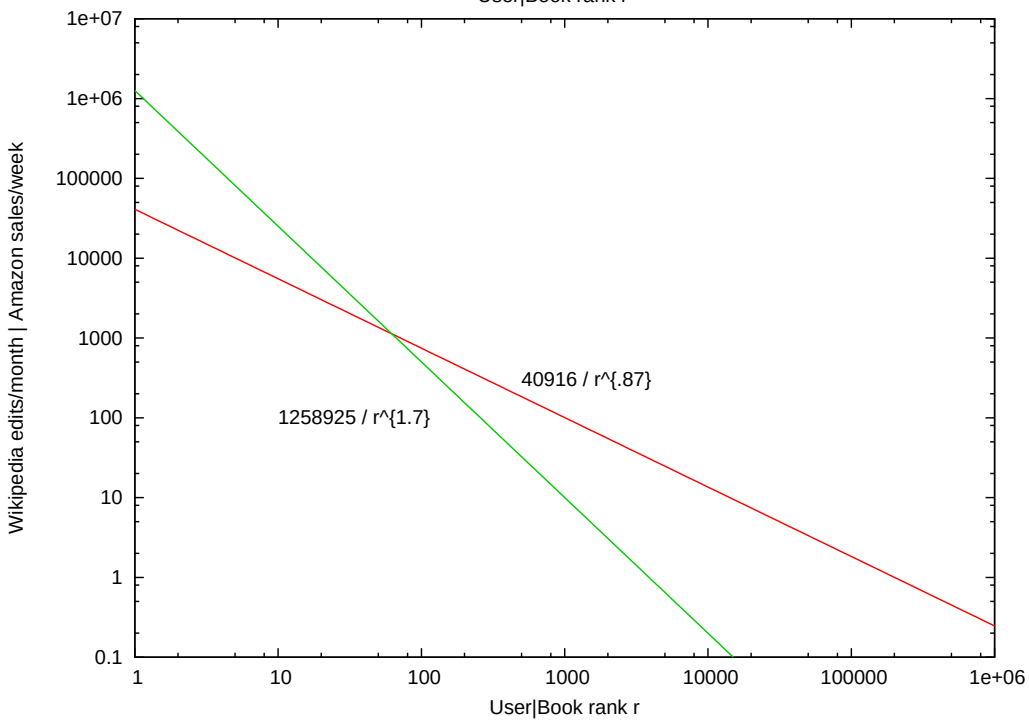


Normalization given by the roughly
1 sale/week for the
200,000th ranked Amazon title:

$$40916r^{-.87}$$

and by the
10 edits/month for the
1000th ranked Wikipedia editor:

$$1258925r^{-1.7}$$



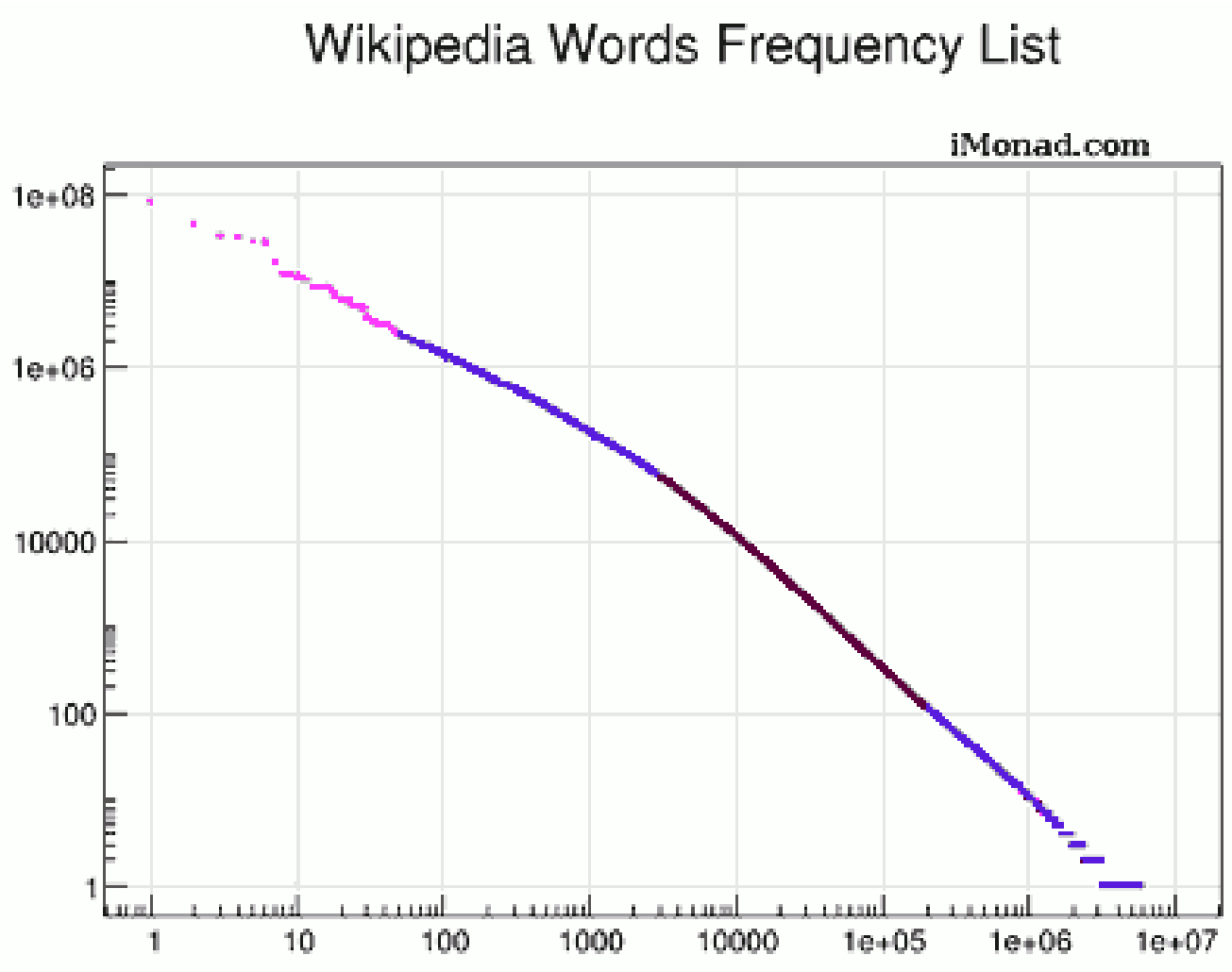
Long tail: about a quarter of
Amazon book sales estimated
to come from the long tail,
i.e., those outside the top
100,000 bestselling titles

Another Wikipedia count (15 May 2010)

`http://imonad.com/seo/wikipedia-word-frequency-list/`

All articles in the English version of Wikipedia, 21GB in XML format (five hours to parse entire file, extract data from markup language, filter numbers, special characters, extract statistics):

- Total tokens (words, no numbers): $T = 1{,}570{,}455{,}731$
- Unique tokens (words, no numbers): $M = 5{,}800{,}280$



"Word frequency distribution follows Zipf's law"

- rank 1–50 (86M-3M), stop words (the, of, and, in, to, a, is, ...)
- rank 51–3K (2.4M-56K), frequent words (university, January, tea, sharp, ...)
- rank 3K–200K (56K-118), words from large comprehensive dictionaries (officiates, polytonality, neologism, ...)
  above rank 50K mostly Long Tail words
- rank 200K–5.8M (117-1), terms from obscure niches, misspelled words, transliterated words from other languages, new words and non-words (euprosthenops, eurotrochilus, lokottaravada, ...)

Some selected words and associated counts

- Google 197920
- Twitter 894
- domain 111850
- domainer 22
- Wikipedia 3226237
- Wiki 176827
- Obama 22941
- Oprah 3885
- Moniker 4974
- GoDaddy 228

Project Gutenberg (per billion)

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists#Project_Gutenberg

Over 36,000 items (Jun 2011), average of $>$ 50 new e-books / week

http://en.wiktionary.org/wiki/Wiktionary:Frequency_lists/PG/2006/04/1-10000

- the 56271872
- of 33950064
- and 29944184
- to 25956096
- in 17420636
- I 11764797
- that 11073318
- was 10078245
- his 8799755
- he 8397205
- it 8058110
- with 7725512
- is 7557477
- for 7097981
- as 7037543
- had 6139336
- you 6048903
- not 5741803
- be 5662527
- her 5202501

$\ldots 100,000^{\text{th}}$