Processing Text

406.424 Internet Applications

Jonghun Park

jonghun@snu.ac.kr

Dept. of Industrial Eng.

Seoul National University

9/1/2010

processing text

- converting documents to **index terms**
  - through tokenization, stopping, stemming, …
- why?
  - matching the exact string of characters typed by the user is too restrictive
    - i.e., it doesn't work very well in terms of effectiveness
  - not all words are of equal value in a search
  - sometimes not clear where words begin and end
    - not even clear what a word is in some languages
      - e.g., Chinese, Korean
- much of the meaning of text is captured by counts of word **occurrences** and **co-occurrences**

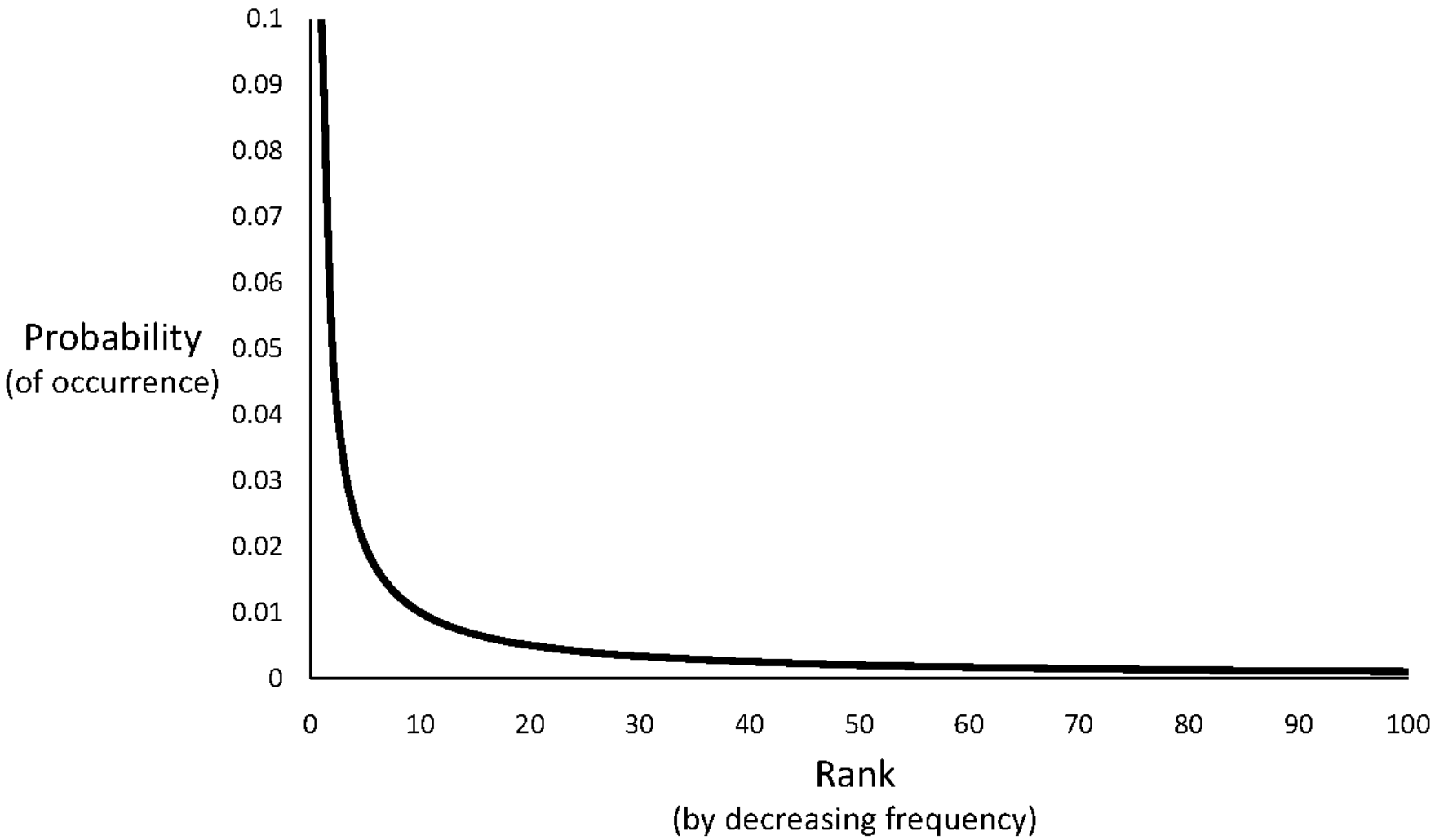
text statistics

- huge variety of words used in text
- but, many statistical characteristics of word occurrences are predictable
  - e.g., distribution of word counts
- retrieval models and ranking algorithms depend heavily on statistical properties of words
  - e.g., important words occur often in documents but are not high frequency in collection (observed by Luhn, 1958)

Zipf's law

- distribution of word frequencies is very **skewed**
  - a few words occur very often, many words hardly ever occur
  - e.g., two most common words ("the", "of") make up about 10% of all word occurrences in text documents
- Zipf's "law":
  - observation that rank (r) of a word times its frequency (f) is approximately a constant (k)
    - assuming words are ranked in order of decreasing frequency
  - i.e., $r \times f = k$ or $r \times P_r = c$, where P_r is probability of word occurrence and $c \approx 0.1$ for English

Zipf's law


0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
Probability
(of occurrence)
0
10
20
30
40
50
60
70
80
90
100
Rank
(by decreasing frequency)

news collection (AP89) statistics

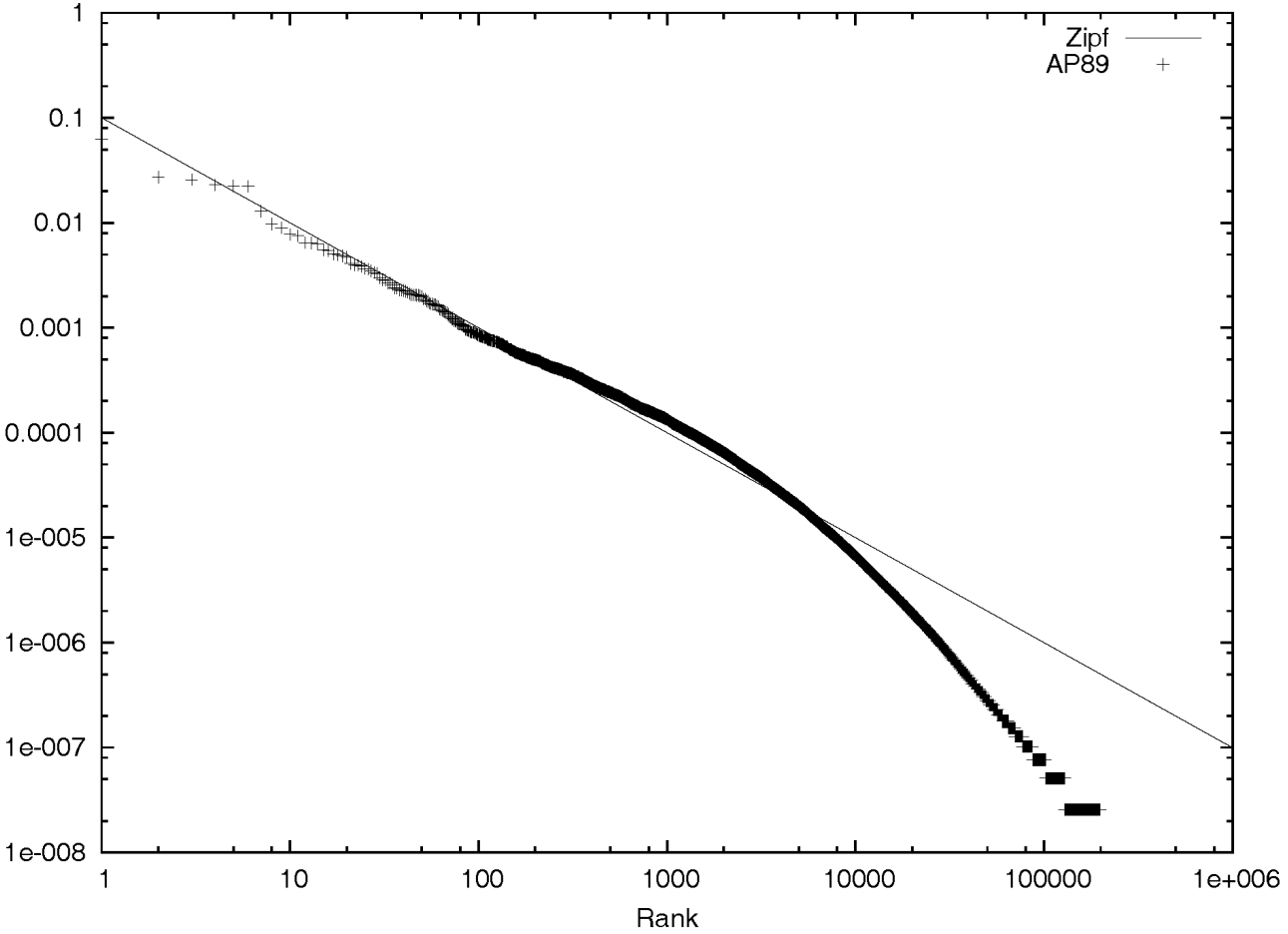
total documents	84,678
total word occurrences	39,749,179
vocabulary size	198,763
words occurring > 1000 times	4,169
words occurring once	70,064

Word	*Freq*	*r*	P_r(%)	$r \times P_r$
assistant	5,095	1,021	.013	0.13
sewers	100	17,110	2.56×10^{-4}	0.04
toothbrush	10	51,555	2.56×10^{-5}	0.01
hazmat	1	166,945	2.56×10^{-6}	0.04

top 50 words from AP89

Word	*Freq.*	*r*	P_r(%)	$r.P_r$	*Word*	*Freq*	*r*	P_r(%)	$r.P_r$
the	2,420,778	1	6.49	0.065	has	136,007	26	0.37	0.095
of	1,045,733	2	2.80	0.056	are	130,322	27	0.35	0.094
to	968,882	3	2.60	0.078	not	127,493	28	0.34	0.096
a	892,429	4	2.39	0.096	who	116,364	29	0.31	0.090
and	865,644	5	2.32	0.120	they	111,024	30	0.30	0.089
in	847,825	6	2.27	0.140	its	111,021	31	0.30	0.092
said	504,593	7	1.35	0.095	had	103,943	32	0.28	0.089
for	363,865	8	0.98	0.078	will	102,949	33	0.28	0.091
that	347,072	9	0.93	0.084	would	99,503	34	0.27	0.091
was	293,027	10	0.79	0.079	about	92,983	35	0.25	0.087
on	291,947	11	0.78	0.086	i	92,005	36	0.25	0.089
he	250,919	12	0.67	0.081	been	88,786	37	0.24	0.088
is	245,843	13	0.65	0.086	this	87,286	38	0.23	0.089
with	223,846	14	0.60	0.084	their	84,638	39	0.23	0.089
at	210,064	15	0.56	0.085	new	83,449	40	0.22	0.090
by	209,586	16	0.56	0.090	or	81,796	41	0.22	0.090
it	195,621	17	0.52	0.089	which	80,385	42	0.22	0.091
from	189,451	18	0.51	0.091	we	80,245	43	0.22	0.093
as	181,714	19	0.49	0.093	more	76,388	44	0.21	0.090
be	157,300	20	0.42	0.084	after	75,165	45	0.20	0.091
were	153,913	21	0.41	0.087	us	72,045	46	0.19	0.089
an	152,576	22	0.41	0.090	percent	71,956	47	0.19	0.091
have	149,749	23	0.40	0.092	up	71,082	48	0.19	0.092
his	142,285	24	0.38	0.092	one	70,266	49	0.19	0.092
but	140,880	25	0.38	0.094	people	68,988	50	0.19	0.093

Zipf's law for AP89



- note the problems at high and low frequencies

Zipf's law

- what is the proportion of words with a given frequency?
  - word that occurs n times has rank $r_n = k/n$
  - # of words with frequency n is
    - $r_n - r_{n+1} = k/n - k/(n+1) = k/n(n+1)$
  - proportion found by dividing by total # of words
    - total # of words = the rank of the last word in the vocabulary = $k/1 = k$
  - so, proportion with frequency n is $1/n(n+1)$

Zipf's law

- example word frequency ranking

Rank	*Word*	*Frequency*
1000	concern	5,100
1001	spoke	5,100
1002	summit	5,100
1003	bring	5,099
1004	star	5,099
1005	immediate	5,099
1006	chemical	5,099
1007	african	5,098

- to compute number of words with frequency 5,099
  - rank of "chemical" minus the rank of "summit"
  - $1006 - 1002 = 4$

example

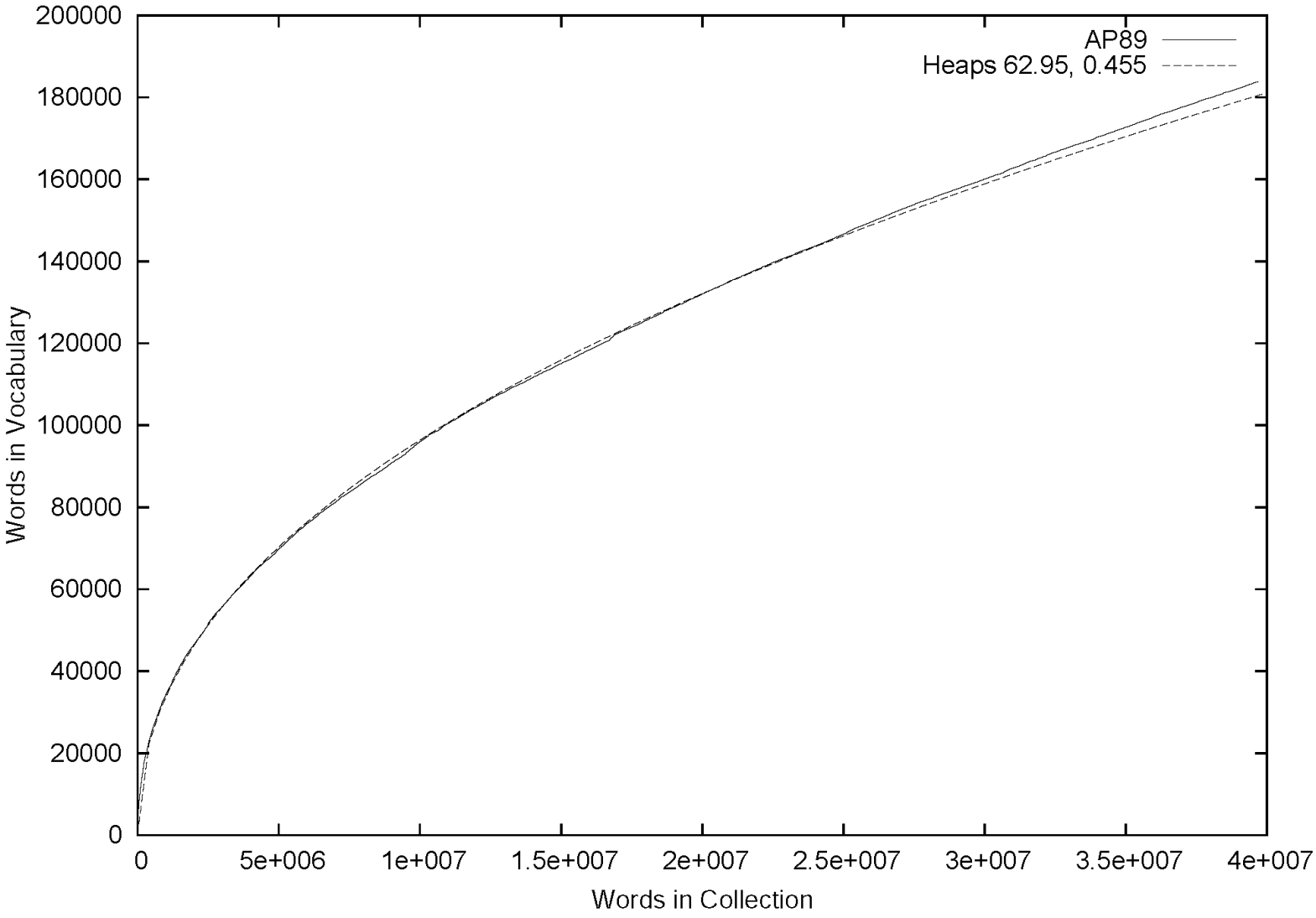
Number of Occurrences (n)	*Predicted Proportion (1/n(n+1))*	*Actual Proportion*	*Actual Number of Words*
1	.500	.402	204,357
2	.167	.132	67,082
3	.083	.069	35,083
4	.050	.046	23,271
5	.033	.032	16,332
6	.024	.024	12,421
7	.018	.019	9,766
8	.014	.016	8,200
9	.011	.014	6,907
10	.009	.012	5,893

- proportions of words occurring *n* times in 336,310 TREC documents
- vocabulary size is 508,209

vocabulary growth

- as corpus grows, so does vocabulary size
  - fewer new words when corpus is already large
- observed relationship (Heaps' law):
  - $v = k \times n^{\beta}$
  - where v is vocabulary size (number of unique words)
  - n is the number of words in corpus
  - k, β are parameters that vary for each corpus (typical values given are $10 \leq k \leq 100$ and $\beta \approx 0.5$)

AP89 Example

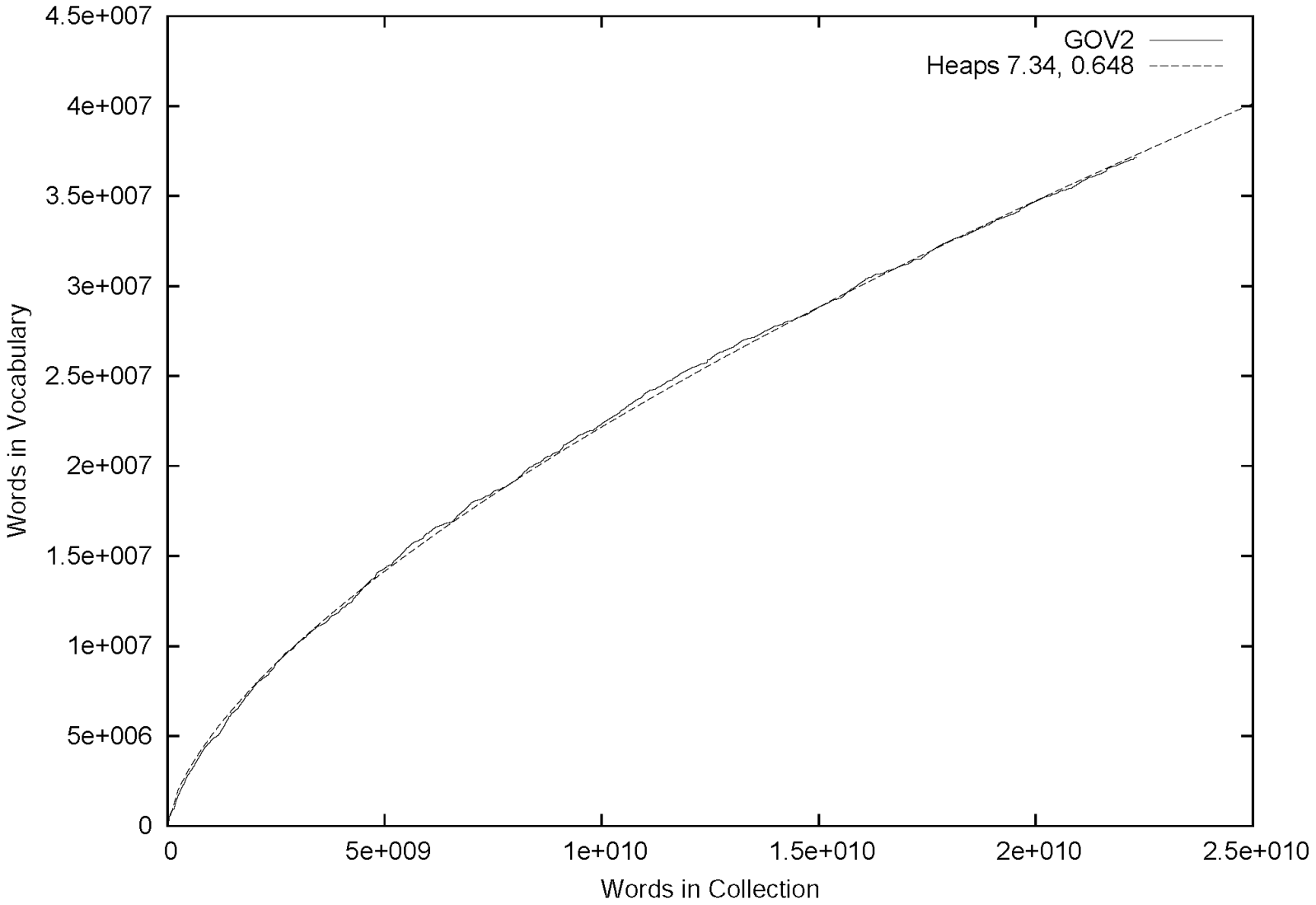


$k = 62.95, \beta = 0.455$

Heaps' law predictions

- predictions for TREC collections are accurate for large numbers of words
  - e.g., first 10,879,522 words of the AP89 collection scanned
  - prediction is 100,151 unique words
  - actual number is 100,024
- predictions for small numbers of words (i.e. < 1000) are much worse

GOV2 (web) example



web example

- Heaps' law works with very large corpora
  - new words occurring even after seeing 30 million
  - parameter values different than typical TREC values
- new words come from a variety of sources
  - spelling errors, invented words (e.g. product, company names), code, other languages, email addresses, etc.
- search engines must deal with these large and growing vocabularies

estimating result set size

- how many pages contain **all** of the query terms?
- for the query "a b c":
  - $f_{abc} = N \cdot f_a/N \cdot f_b/N \cdot f_c/N = (f_a \cdot f_b \cdot f_c)/N^2$
  - assuming that terms occur independently
    - note: $P(a\ )\ b\ )\ \ c) = P(a) \cdot P(b) \cdot P(c)$
  - f_{abc} is the estimated size of the result set
  - f_a, f_b, f_c are the numbers of documents that terms *a*, *b*, and *c* occur in
  - N is the number of documents in the collection

tropical fish aquarium Search

Web results Page 1 of 3,880,000 results

GOV2 example

Word(s)	*Document Frequency*	*Estimated Frequency*
tropical	120,990	
fish	1,131,855	
aquarium	26,480	
breeding	81,885	
tropical fish	18,472	5,433
tropical aquarium	1,921	127
tropical breeding	5,510	393
fish aquarium	9,722	1,189
fish breeding	36,427	3,677
aquarium breeding	1,848	86
tropical fish aquarium	1,529	6
tropical fish breeding	3,629	18

collection size (N) is 25,205,179

result set size estimation

- poor estimates because words are **not independent**
- better estimates possible if **co-occurrence** information available
- $P(a \cap b \cap c) = P(a \cap b) \cdot P(c \mid (a \cap b))$
  $\leq P(a \cap b) \cdot P(c \mid a)$ and $\leq P(a \cap b) \cdot P(c \mid b)$
- $f_{tropical \cap aquarium \cap fish} = f_{tropical \cap aquarium} \cdot f_{fish \cap aquarium} / f_{aquarium}$
  $= 1921 \cdot 9722/26480 = 705$
- $f_{tropical \cap breeding \cap fish} = f_{tropical \cap breeding} \cdot f_{fish \cap breeeding} / f_{breeding}$
  $= 5510 \cdot 36427/81885 = 2451$

result set estimation

- even better estimates using **initial result set**
  - estimate is simply *C*/*s*
    - *s* is the **proportion** of the total documents that have been ranked
    - *C* is the number of documents found that contain **all the query words**
  - *s* is measured by the proportion of the documents containing **the least frequent word** that have been processed, since all results must contain that word
  - e.g., “tropical fish aquarium” in GOV2
    - after processing 3,000 out of the total 26,480 documents that contain “aquarium”, assume that the number of documents containing all 3 words, $C = 258$
      $f_{tropical \cap fish \cap aquarium} = 258/(3000 \div 26480) = 2{,}277$
    - after processing 20% of the documents,
      $f_{tropical \cap fish \cap aquarium} = 1{,}778$ (1,529 is real value)

estimating collection size

- important issue for web search engines
- simple technique: use independence model
  - given two words *a* and *b* that are **independent**
    $f_{ab}/N = f_a/N \cdot f_b/N$
    $N = (f_a \cdot f_b)/f_{ab}$
  - e.g., for GOV2
    $f_{lincoln} = 771{,}326$ $f_{tropical} = 120{,}990$ $f_{lincoln \cap tropical} = 3{,}018$
    $N = (120990 \cdot 771326)/3018 = 30{,}922{,}045$
    (actual number is 25,205,179)

tokenizing

- forming **words** from **sequence of characters**
- surprisingly complex in English, can be harder in other languages
- early IR systems:
  - any sequence of alphanumeric characters of length 3 or more
  - terminated by a space or other special character
  - upper-case changed to lower-case

tokenizing

- example:
  - “Bigcorp's 2007 bi-annual report showed profits rose 10%.” => “bigcorp 2007 annual report showed profits rose”
- too simple for search applications or even large-scale experiments
  - too much information lost
  - small decisions in tokenizing can have major impact on effectiveness of some queries

tokenizing problems

- small words can be important in some queries, usually in combinations
  - xp, ma, pm, ben e king, el paso, master p, gm, j lo, world war II
- both hyphenated and non-hyphenated forms of many words are common
  - sometimes hyphen is not needed
    - e-bay, wal-mart, active-x, cd-rom, t-shirts
  - at other times, hyphens should be considered either as part of the word or a word separator
    - winston-salem, mazda rx-7, e-cards, pre-diabetes, t-mobile, spanish-speaking

tokenizing problems

- special characters are an important part of tags, URLs, code in documents
- capitalized words can have different meaning from lower case words
  - Bush, Apple
- apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - rosie o'donnell, can't, don't, 80's, 1890's, men's straw hats, master's degree, england's ten largest cities, shriner's

tokenizing problems

- **numbers** can be important, including decimals
  - nokia 3250, top 10 courses, united 93, quicktime 6.5 pro, 92.3 the beat, 288358
- **periods** can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - I.B.M., Ph.D., imlab.snu.ac.kr, F.E.A.R.
- note: **tokenizing steps for queries must be identical to steps for documents**

tokenizing process

- first step is to use parser to identify appropriate parts of document to tokenize
- defer complex decisions to other components
  - word is any sequence of alphanumeric characters, terminated by a space or special character, with everything converted to lower-case
  - everything indexed
  - example: 92.3 → 92 3 but search finds documents with 92 and 3 adjacent
  - incorporate some rules
    - apostrophes in words ignored: o'connor → oconnor bob's → bobs
    - periods in abbreviations ignored: I.B.M. → ibm Ph.D. → ph d

stopping

- function words (determiners, prepositions) have little meaning on their own
  - the, a, an, that, …
- high occurrence frequencies
- treated as **stopwords** (i.e. removed)
  - reduce index space, improve response time, improve effectiveness
- can be important in combinations
  - e.g., “to be or not to be”

stopping

- **stopword** list can be created from high-frequency words or based on a standard list
- lists are customized for applications, domains, and even parts of documents
  - e.g., “click” and “here” is a good stopword for anchor text
- best policy is to **index all words** in documents, make decisions about which words to use **at query time**

stemming

- many morphological variations of words
  - inflectional (plurals, tenses)
  - derivational (making verbs nouns etc.)
- in most cases, these have the **same** or very **similar meanings**
- stemmers attempt to **reduce morphological variations** of words to a common stem
  - usually involves removing suffixes
- can be done **at indexing time** or as part of **query processing** (like stopwords)

stemming

- generally a small but significant effectiveness improvement
  - can be crucial for some languages
  - e.g., 5-10% improvement for English, up to 50% in Arabic
- 2 basic types
  - dictionary-based: uses lists of related words
  - algorithmic: uses program to determine related words
- algorithmic stemmers
  - suffix-s stemmer: remove ‘s’ endings assuming plural
    - e.g., cats → cat, lakes → lake, wiis → wii
    - many **false negatives**: supplies → supplie
    - some **false positives**: ups → up

Porter stemmer

- algorithmic stemmer used in IR experiments since the 70s
- consists of **a series of rules** designed to the longest possible suffix at each step
- effective in TREC
- produces **stems not words**
- makes a number of errors and difficult to modify
- http://tartarus.org/martin/PorterStemmer/

Porter stemmer

- example step (1 of 5)

Step 1a:

- Replace ***sses*** by ***ss*** (e.g., stresses → stress).
- Delete ***s*** if the preceding word part contains a vowel not immediately before the ***s*** (e.g., gaps → gap but gas → gas).
- Replace ***ied*** or ***ies*** by ***i*** if preceded by more than one letter, otherwise by ***ie*** (e.g., ties → tie, cries → cri).
- If suffix is ***us*** or ***ss*** do nothing (e.g., stress → stress).

Step 1b:

- Replace ***eed***, ***eedly*** by ***ee*** if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete ***ed***, ***edly***, ***ing***, ***ingly*** if the preceding word part contains a vowel, and then if the word ends in ***at***, ***bl***, or ***iz*** add ***e*** (e.g., fished → fish, pirating → pirate), or if the word ends with a double letter that is not ***ll***, ***ss*** or ***zz***, remove the last letter (e.g., falling→ fall, dripping → drip), or if the word is short, add ***e*** (e.g., hoping → hope).
- Whew!

Krovetz stemmer

- hybrid algorithmic-dictionary
  - word checked in dictionary
    - if present, either left alone or replaced with “exception”
    - if not present, word is checked for suffixes that could be removed
    - after removal, dictionary is checked again
- produces **words** not stems
- comparable effectiveness
- lower false positive rate, somewhat higher false negative

stemmer comparison

Original text:

Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Porter stemmer:

document describ market strategi carri compani agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale market share stimul demand price cut volum sale

Krovetz stemmer:

document describe marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochemic pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

phrases

- many queries are 2-3 word phrases
- phrases are
  - **more precise** than single words
    - e.g., documents containing “black sea” vs. two words “black” and “sea”
  - **less ambiguous**
    - e.g., “big apple” vs. “apple”
- can be difficult for ranking
  - e.g., given query “fishing supplies”, how do we score documents with
    - exact phrase many times, exact phrase just once, individual words in same sentence, same paragraph, whole document, variations on words?
- how are phrases recognized?
  - identify syntactic phrases using a **part-of-speech** (**POS**) tagger
  - use word **n-grams**
  - store word positions in indexes and use **proximity operators** in queries: e.g., testing whether 2 words occur within a specified text widnow

POS tagging

- POS taggers use **statistical models of text** to predict syntactic tags of words
  - example tags:
    - NN (singular noun), NNS (plural noun), VB (verb), VBD (verb, past tense), VBN (verb, past participle), IN (preposition), JJ (adjective), CC (conjunction, e.g., “and”, “or”), PRP (pronoun), and MD (modal auxiliary, e.g., “can”, “will”).
- phrases can then be defined as simple **noun groups**, for example

POS tagging example

Original text:

Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Brill tagger:

Document/NN will/MD describe/VB marketing/NN strategies/NNS carried/VBD out/IN by/IN U.S./NNP companies/NNS for/IN their/PRP agricultural/JJ chemicals/NNS ,/, report/NN predictions/NNS for/IN market/NN share/NN of/IN such/JJ chemicals/NNS ,/, or/CC report/NN market/NN statistics/NNS for/IN agrochemicals/NNS ,/, pesticide/NN ,/, herbicide/NN ,/, fungicide/NN ,/, insecticide/NN ,/, fertilizer/NN ,/, predicted/VBN sales/NNS ,/, market/NN share/NN ,/, stimulate/VB demand/NN ,/, price/NN cut/NN ,/, volume/NN of/IN sales/NNS ./.

example noun phrases

TREC data		Patent data	
Frequency	*Phrase*	*Frequency*	*Phrase*
65824	united states	975362	present invention
61327	article type	191625	u.s. pat
33864	los angeles	147352	preferred embodiment
18062	hong kong	95097	carbon atoms
17788	north korea	87903	group consisting
17308	new york	81809	room temperature
15513	san diego	78458	seq id
15009	orange county	75850	brief description
12869	prime minister	66407	prior art
12799	first time	59828	perspective view
12067	soviet union	58724	first embodiment
10811	russian federation	56715	reaction mixture
9912	united nations	54619	detailed description
8127	southern california	54117	ethyl acetate
7640	south korea	52195	example 1
7620	end recording	52003	block diagram
7524	european union	46299	second embodiment
7436	south africa	41694	accompanying drawings
7362	san francisco	40554	output signal
7086	news conference	37911	first end
6792	city council	35827	second end
6348	middle east	34881	appended claims
6157	peace process	33947	distal end
5955	human rights	32338	cross-sectional view
5837	white house	30193	outer surface

word n-grams

- POS tagging: **too slow** for large collections
- simpler definition: phrase is **any sequence of *n* words** – known as **n-grams**
  - bigram: 2 word sequence, trigram: 3 word sequence, unigram: single words
  - N-grams also used at character level for applications such as OCR
- N-grams typically formed from **overlapping sequences** of words
  - i.e. move n-word “window” one word at a time in document

n-grams

- **frequent n-grams** are more likely to be **meaningful phrases**
- n-grams form a Zipf distribution
  - better fit than words alone
- could index all n-grams up to specified length
  - much faster than POS tagging
  - uses a lot of storage
    - e.g., document containing 1,000 words would contain 3,990 instances of word n-grams of length $2 \leq n \leq 5$

Google n-grams

- web search engines index n-grams
- Google sample:

Number of tokens:	1,024,908,267,229
Number of sentences:	95,119,665,584
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

- most frequent trigram in English is “all rights reserved”

document structure and markup

- some parts of documents are more important than others
- document parser **recognizes structure** using markup, such as HTML tags
  - headers, anchor text, bolded text all likely to be important
  - metadata can also be important
  - links used for **link analysis**

example web page

Tropical fish

From Wikipedia, the free encyclopedia

Tropical fish include fish found in t
both freshwater and salt water specie
refer only those requiring fresh wate
fish.

Tropical fish are popular aquarium f
freshwater fish, this coloration typic
are generally pigmented.

```
<html>
<head>
<meta name="keywords" content="Tropical fish, Airstone, Albinism, Algae eater,
Aquarium, Aquarium fish feeder, Aquarium furniture, Aquascaping, Bath treatment
(fishkeeping),Berlin Method, Biotope" />
…
<title>Tropical fish - Wikipedia, the free encyclopedia</title>
</head>
<body>
 …
<h1 class="firstHeading">Tropical fish</h1>
…
<p><b>Tropical fish</b> include <a href="/wiki/Fish" title="Fish">fish</a> found in <a
href="/wiki/Tropics" title="Tropics">tropical</a> environments around the world,
including both <a href="/wiki/Fresh_water" title="Fresh water">freshwater</a> and <a
href="/wiki/Sea_water" title="Sea water">salt water</a> species. <a
href="/wiki/Fishkeeping" title="Fishkeeping">Fishkeepers</a> often use the term
<i>tropical fish</i> to refer only those requiring fresh water, with saltwater tropical fish
referred to as <i><a href="/wiki/List_of_marine_aquarium_fish_species" title="List of
marine aquarium fish species">marine fish</a></i>.</p>
<p>Tropical fish are popular <a href="/wiki/Aquarium" title="Aquarium">aquarium</a>
fish , due to their often bright coloration. In freshwater fish, this coloration typically
derives from <a href="/wiki/Iridescence" title="Iridescence">iridescence</a>, while salt
water fish are generally <a href="/wiki/Pigment" title="Pigment">pigmented</a>.</p>
…
</body></html>
```


link analysis

- links are a key component of the web
- important for navigation, but also for search
  - e.g., Example website
  - “Example website” is the **anchor text**
  - “http://example.com” is the destination link
  - both are used by search engines

anchor text

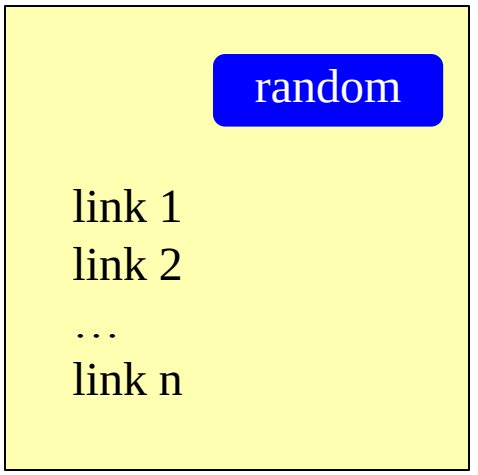
- used as a **description of the content of the destination page**
  - i.e., collection of anchor text in all links pointing to a page used as an additional text field
- anchor text tends to be short, descriptive, and similar to query text
- retrieval experiments have shown that anchor text has significant impact on effectiveness for some types of queries
  - i.e., more than PageRank

PageRank

- billions of web pages, some more informative than others
- links can be viewed as information about the **popularity** (authority?) of a web page
  - can be used by ranking algorithm
- **inlink count** could be used as simple measure
- link analysis algorithms like PageRank provide more reliable ratings
  - less susceptible to link spam

random surfer model

- browse the web using the following algorithm:
  - choose a random number *r* between 0 and 1
  - if $r < \lambda$:
    - go to a random page
  - if $r \geq \lambda$:
    - click a link at random on the current page
  - start again

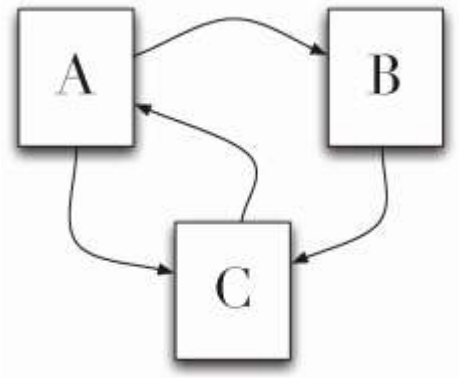


- PageRank of a page is **the probability that the "random surfer" will be looking at that page**
  - links from popular pages will increase PageRank of pages they point to

dangling links

- random jump prevents getting stuck on pages that
  - do not have links
  - contains only links that no longer point to other pages
  - have links forming a loop
- links that point to the first two types of pages are called **dangling links**
  - may also be links to pages that have not yet been crawled

PageRank



- PageRank (PR) of page C = PR(A)/2 + PR(B)/1
- more generally,

$$PR(u) = \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

  - where B_u is the set of pages that **point to *u***, and L_v is the number of outgoing links from page *v* (not counting duplicate links)

PageRank

- don't know PageRank values at start
- assume equal values (1/3 in this case), then iterate:
  - initial: $PR(C) = 0.33$, $PR(A) = 0.33$, $PR(B) = 0.33$
  - first iteration: $PR(C) = 0.33/2 + 0.33 = 0.5$, $PR(A) = 0.33$, and $PR(B) = 0.17$
  - second: $PR(C) = 0.33/2 + 0.17 = 0.33$, $PR(A) = 0.5$, $PR(B) = 0.17$
  - third: $PR(C) = 0.42$, $PR(A) = 0.33$, $PR(B) = 0.25$
- **converges** to $PR(C) = 0.4$, $PR(A) = 0.4$, and $PR(B) = 0.2$

PageRank

- taking random page jump into account, 1/3 chance of going to any page **when $r < \lambda$**
- $PR(\mathrm{C}) = \lambda/3 + (1 - \lambda) \cdot (PR(\mathrm{A})/2 + PR(\mathrm{B})/1)$
  - note: chance of randomly visiting one of 3 pages is λ
- more generally,

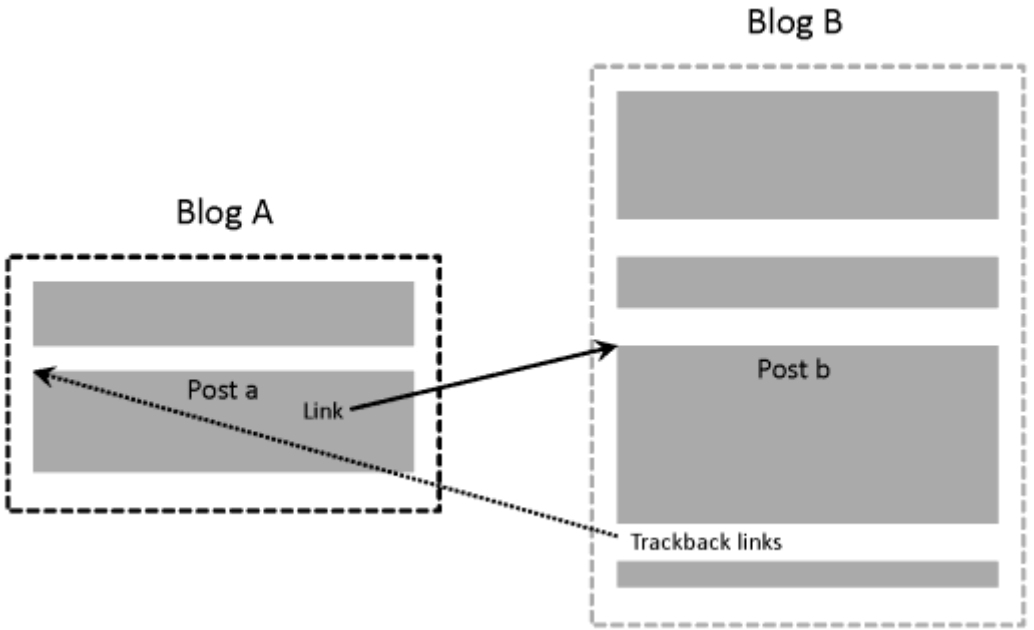
$$PR(u) = \frac{\lambda}{N} + (1 - \lambda). \sum_{v \in B_u} \frac{PR(v)}{L_v}$$

  - where N is the number of pages, λ typically 0.15

```
1: procedure PageRank(G)
2:     ▷ G is the web graph, consisting of vertices (pages) and edges (links).
3:     (P, L) ← G                                  ▷ Split graph into pages and links
4:     I ← a vector of length |P|                  ▷ The current PageRank estimate
5:     R ← a vector of length |P|   ▷ The resulting better PageRank estimate
6:     for all entries I_i ∈ I do
7:         I_i ← 1/|P|                 ▷ Start with each page being equally likely
8:     end for
9:     while R has not converged do
10:        for all entries R_i ∈ R do
11:            R_i ← λ/|P| ▷ Each page has a λ/|P| chance of random selection
12:        end for
13:        for all pages p ∈ P do
14:            Q ← the set of pages such that (p, q) ∈ L and q ∈ P
15:            if |Q| > 0 then
16:                for all pages q ∈ Q do
17:                    R_q ← R_q + (1 − λ)I_p/|Q|      ▷ Probability I_p of being at
   page p
18:                end for
19:            else
20:                for all pages q ∈ P do
21:                    R_q ← R_q + (1 − λ)I_p/|P|
22:                end for
23:            end if
24:            I ← R                       ▷ Update our current PageRank estimate
25:        end for
26:    end while
27:    return R
28: end procedure
```


link quality

- link quality is affected by spam and other factors
  - **link farms** to increase PageRank
  - **trackback links** in blogs can create loops
  - preventing promotion of other web sites by posting links to them in the comments of popular blogs
    - e.g., “Come visit my <a **`rel=nofollow`** href="http://www.page.com">web page.”



information extraction

- automatically extract **structure** from text
  - annotate document using tags to identify extracted structure
- named entity recognition
  - named entity: words that refer to something of interest in a particular application
  - e.g., people, companies, locations, dates, product names, prices, etc.

named entity recognition

Fred Smith, who lives at 10 Water Street, Springfield, MA, is a long-time collector of **tropical fish.**

<PersonName><GivenName>Fred</GivenName> <Sn>Smith</Sn> </PersonName>, who lives at <address><Street >10 Water Street</Street>, <City>Springfield</City>, <State>MA</State></address>, is a long-time collector of tropical fish.

- example showing **semantic annotation** of text using XML tags
- information extraction also includes document structure and more complex features such as **relationships** and **events**

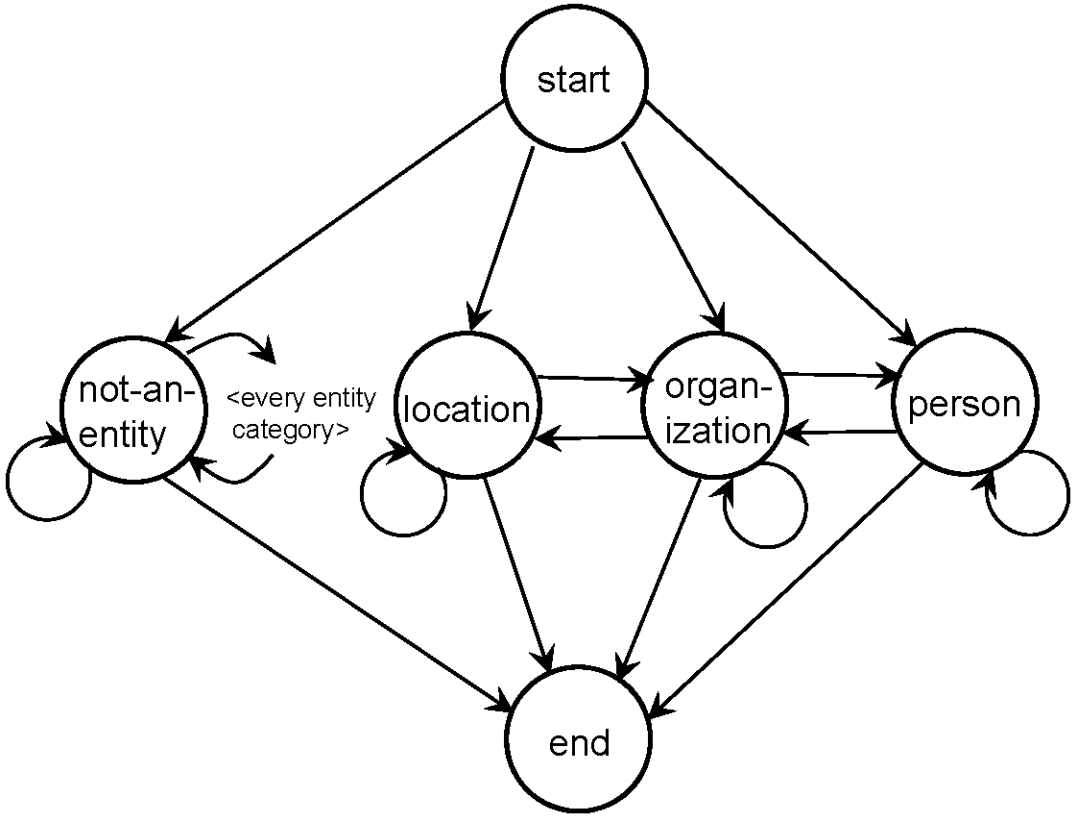
named entity recognition

- rule-based
  - rules either developed **manually** by trial and error or using **machine learning** techniques
  - uses lexicons (lists of words and phrases) that categorize names
    - e.g., locations, peoples' names, organizations, etc.
  - rules also used to verify or find new entity names
    - "<number> <word> street" for addresses
    - "<street address>, <city>" or "in <city>" to verify city names
    - "<street address>, <city>, <state>" to find new cities
    - "<title> <name>" to find new names
  - e.g., http://gate.ac.uk/
- Statistical
  - uses a probabilistic model of the words in and around an entity
  - probabilities estimated using **training data** (manually annotated text)
  - Hidden Markov Model (HMM) is one approach

HMM for extraction

- resolve ambiguity in a word using **context**
  - e.g., “marathon” is a location or a sporting event, “boston marathon” is a specific sporting event
- model context using a **generative model** of the sequence of words
  - **Markov property**: the next word in a sequence depends only on a **small number of the previous words**
- **Markov model** describes a process as a collection of states with transitions between them
  - each transition has a probability associated with it
  - next state depends only on current state and transition probabilities
- **Hidden Markov Model**
  - each state has a set of possible outputs
  - outputs have probabilities

HMM Sentence Model



- each state is associated with a probability distribution over words (the output)
- the words in a sentence are assumed to be either part of an entity name or not part of one

HMM for extraction

- to recognize named entities, find **sequence of "labels"** that give **highest probability for the sentence**
  - only the outputs (words) are visible or observed
  - states are "hidden"
  - e.g., sequence of states with the highest probability might be: <start><name><not-an-entity><location><not-an-entity><end>
- Viterbi algorithm used for recognition
  - requires training data

named entity recognition

- accurate recognition requires about 1M words of training data (1,500 news stories)
  - may be more expensive than developing rules for some applications
- both rule-based and statistical can achieve about 90% effectiveness for categories such as names, locations, organizations
  - others, such as product name, can be much worse

internationalization

- 2/3 of the web is in English
- about 50% of web users do not use english as their primary language
- many (maybe most) search applications have to deal with multiple languages
  - monolingual search: search in one language, but with many possible languages
  - cross-language search: search in multiple languages at the same time

internationalization

- many aspects of search engines are language-neutral
- major differences:
  - text encoding (converting to Unicode)
  - tokenizing (many languages have no word separators)
  - stemming
- cultural differences may also impact interface design and features provided

1. Original text

旱灾在中国造成的影响

(the impact of droughts in China)

2. Word segmentation

旱灾 在 中国 造成 的 影响

drought at china make impact

3. Bigrams

旱灾 灾在 在中 中国 国造
造成 成的 的影 影响