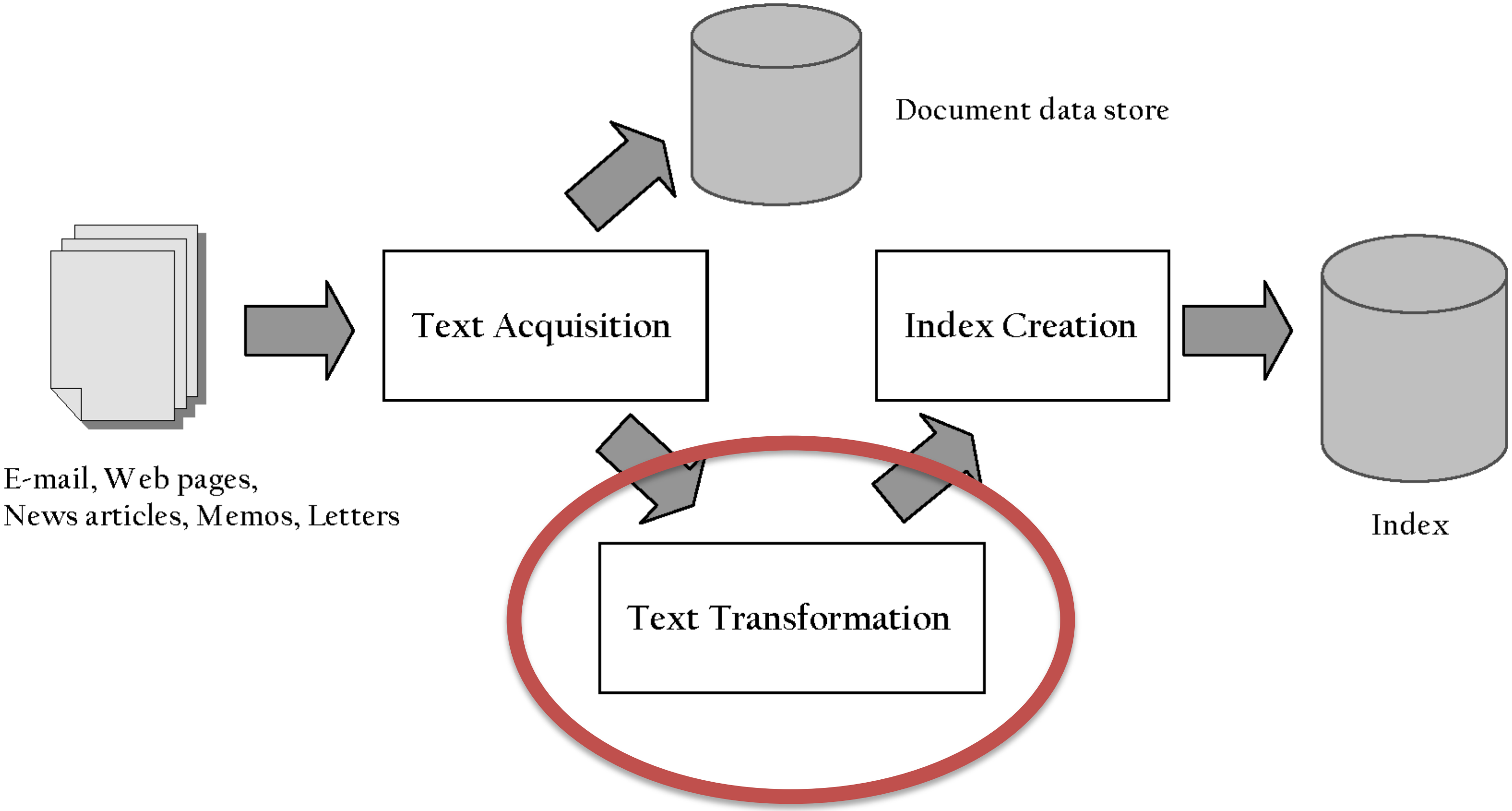
CS6200/IS4200
Information Retrieval

David Smith

Khoury College of Computer Sciences

Northeastern University

Indexing Process



Processing Text

- Converting documents to *index terms*
- 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
    - Linguistic disagreement about what a word is in some languages
      - e.g., Chinese, Korean

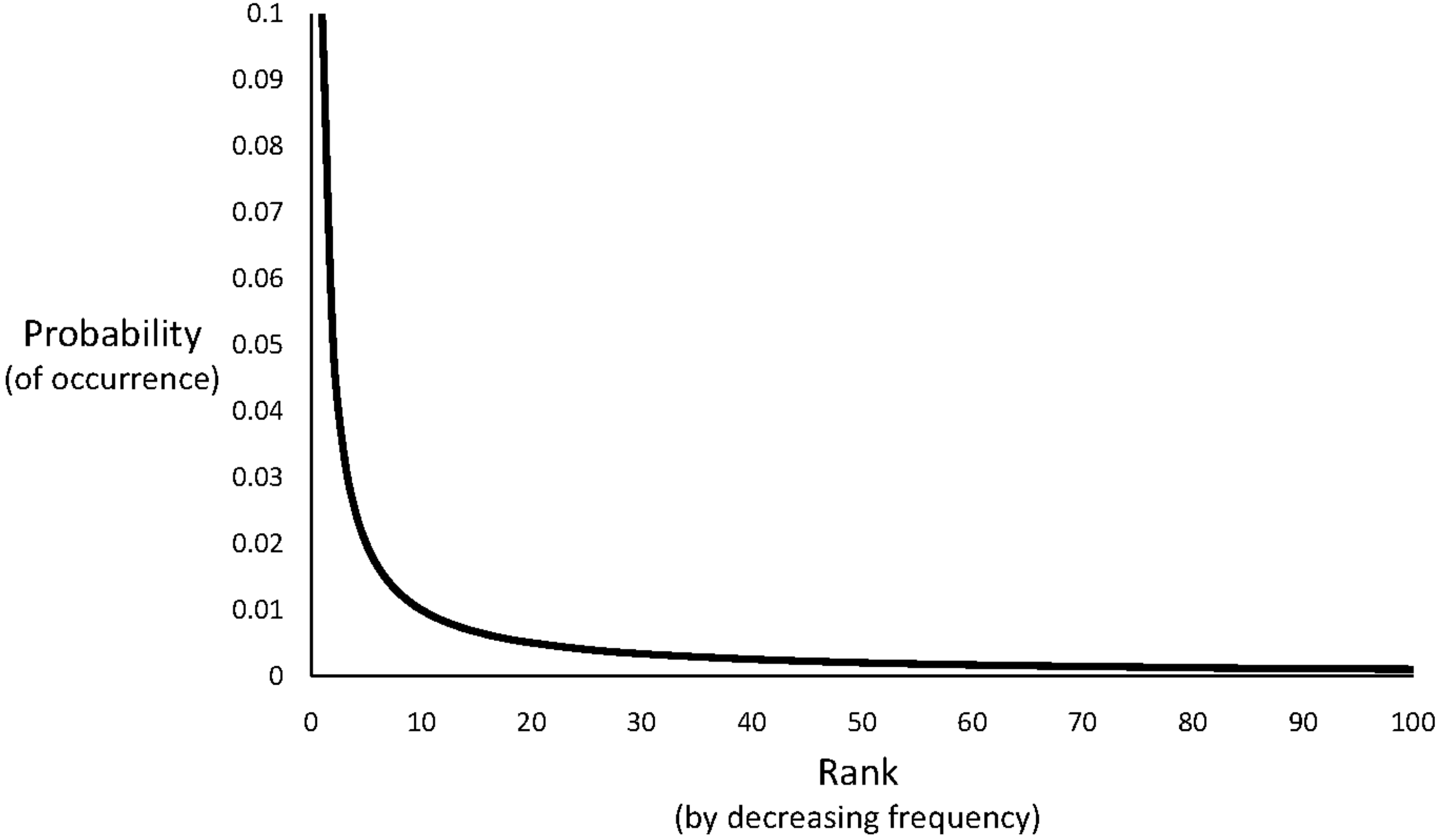
Text Statistics

- Huge variety of words used in text <u>but</u>
- Many statistical characteristics of word occurrences are predictable
  - e.g., distribution of word and n-gram 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

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 (more generally, a "power 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., $r.f \approx \mathrm{k}$ or $r.P_r \approx c$, where P_r is probability of word occurrence and $c \approx 0.1$ for English

Zipf’s Law

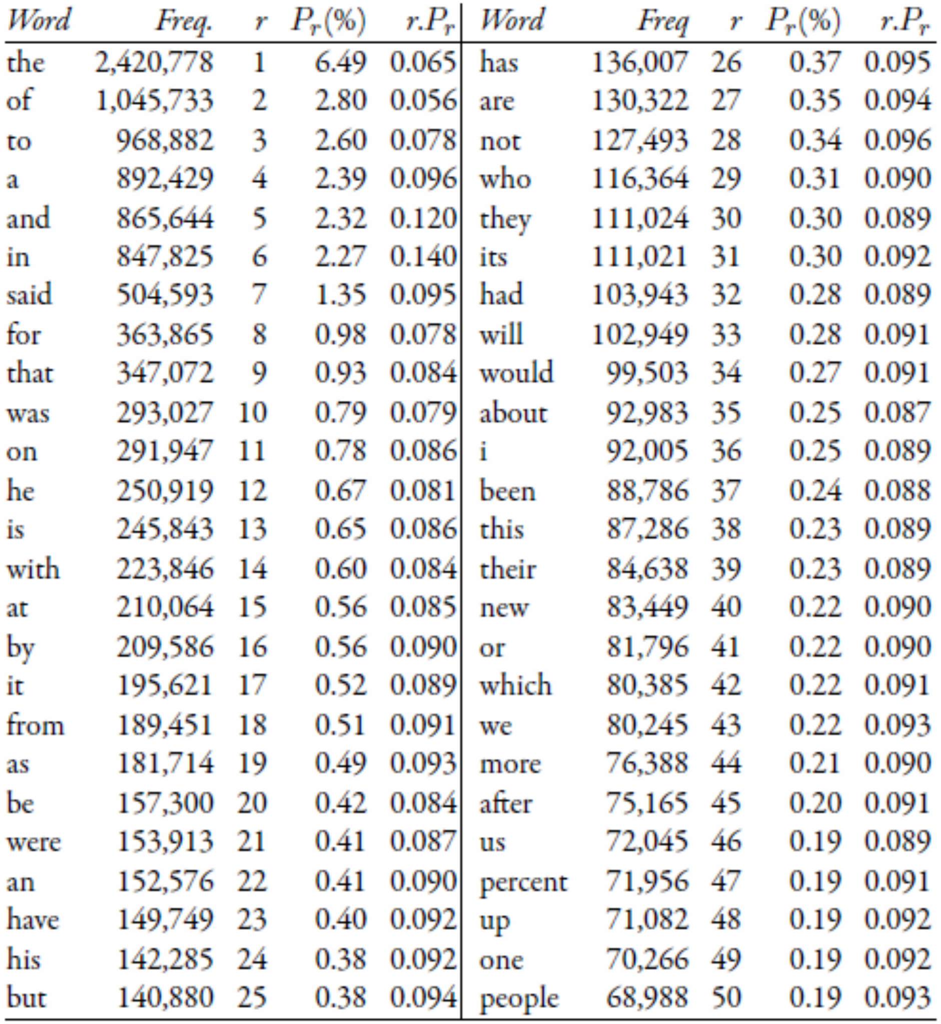

Probability
(of occurrence)
0.1
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0
0
10
20
30
40
50
60
70
80
90
100
Rank
(by decreasing frequency)

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

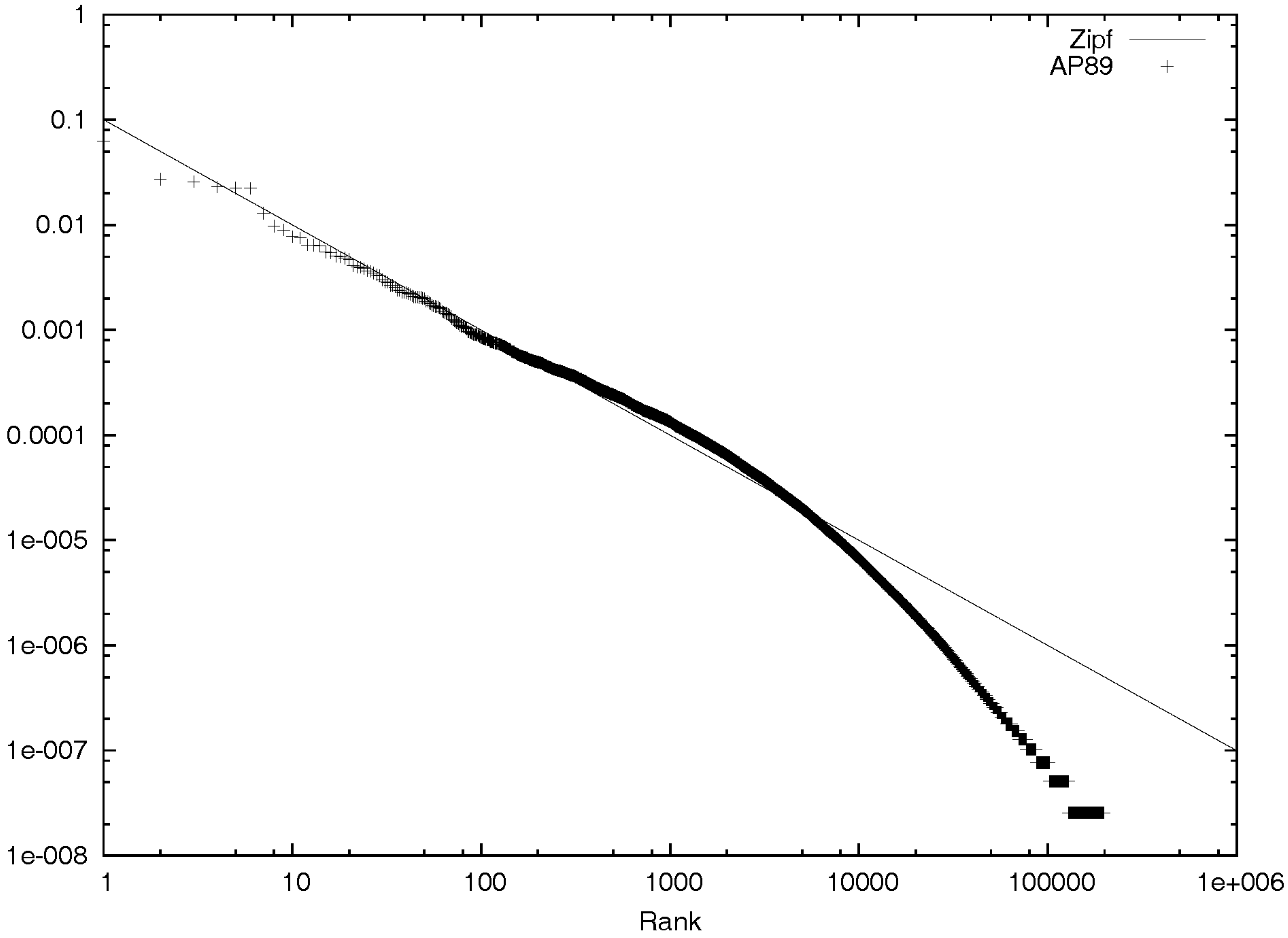
Word	*Freq.*	*r*	*Pr(%)*	*r.Pr*
assistant	5,095	1,021	.013	0.13
sewers	100	17,110	2.56 × 10−4	0.04
toothbrush	10	51,555	2.56 × 10−5	0.01
hazmat	1	166,945	2.56 × 10−6	0.04

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



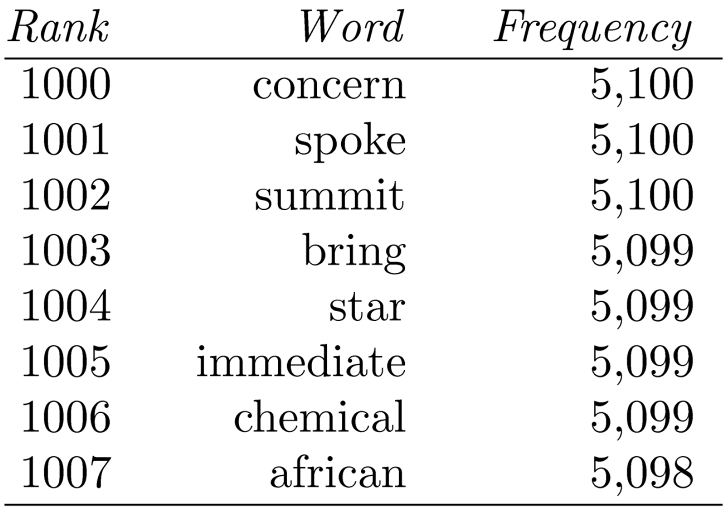
- Log-log plot: Note problems at high and low frequencies

Zipf’s Law

- What is the proportion of words with a given frequency?
  - Word that occurs *n* times has rank $r_n = k/n$
  - Number 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 number of words = highest rank = *k*
  - So, proportion with frequency *n* is $1/n(n+1)$

Zipf’s Law

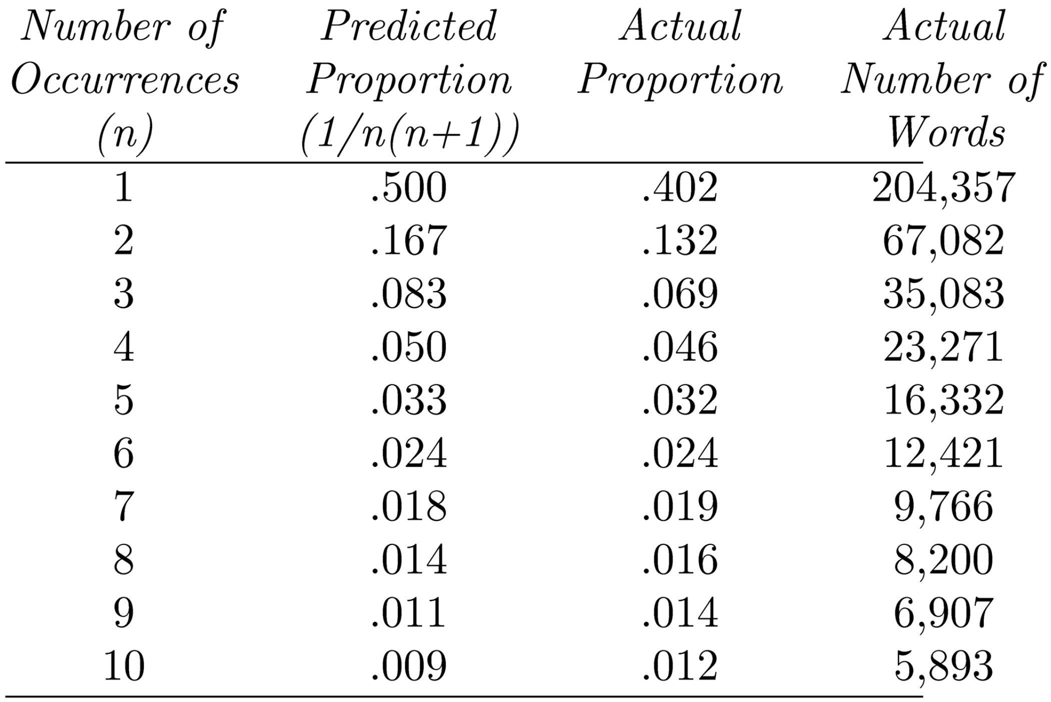
- Example word frequency ranking



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

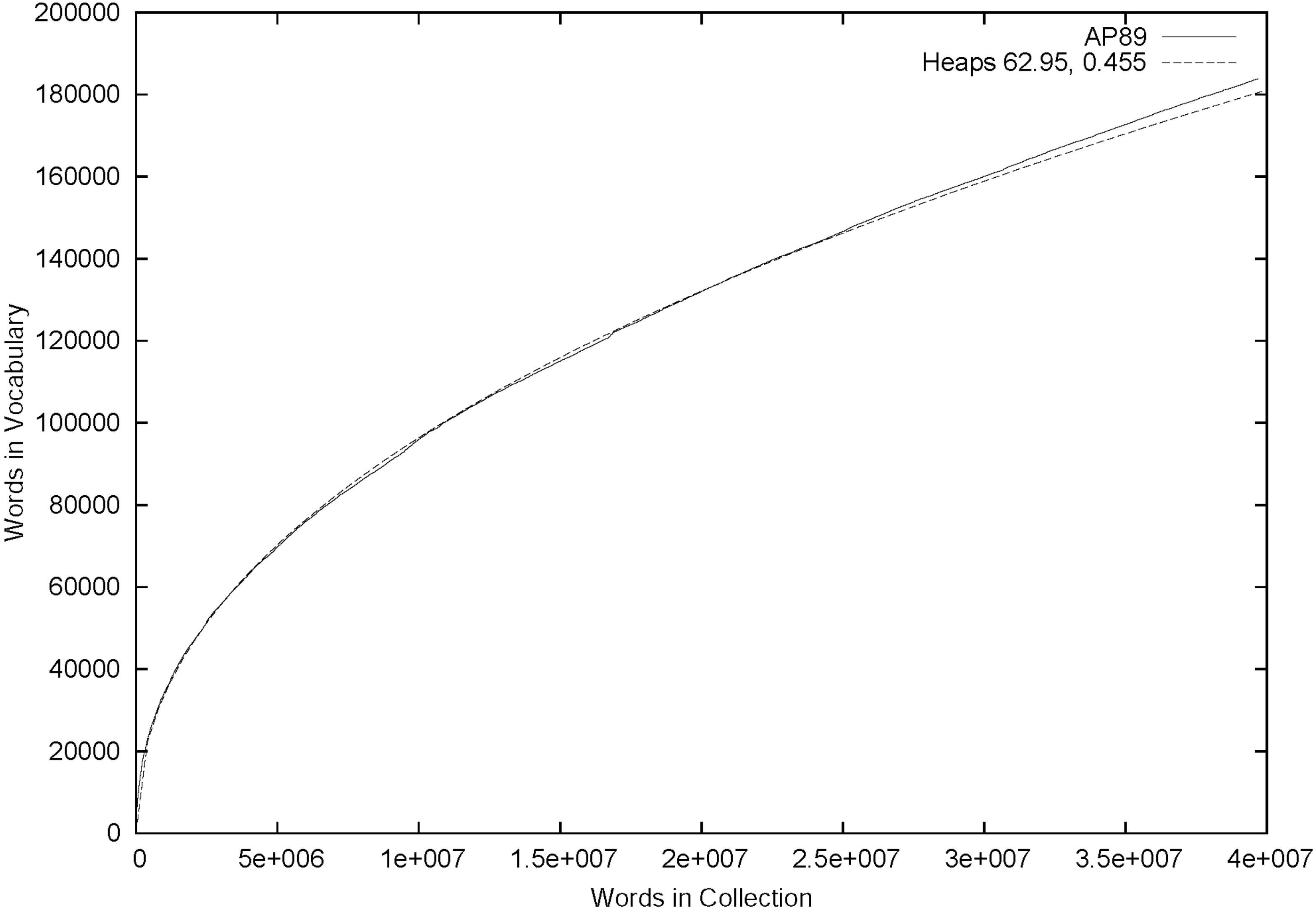
Vocabulary Growth

- As corpus grows, so does vocabulary size
  - Fewer new words when corpus is already large
- Observed relationship (*Heaps' Law*):

$$v = k.n^{\beta}$$

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 $\beta \approx 0.5$)

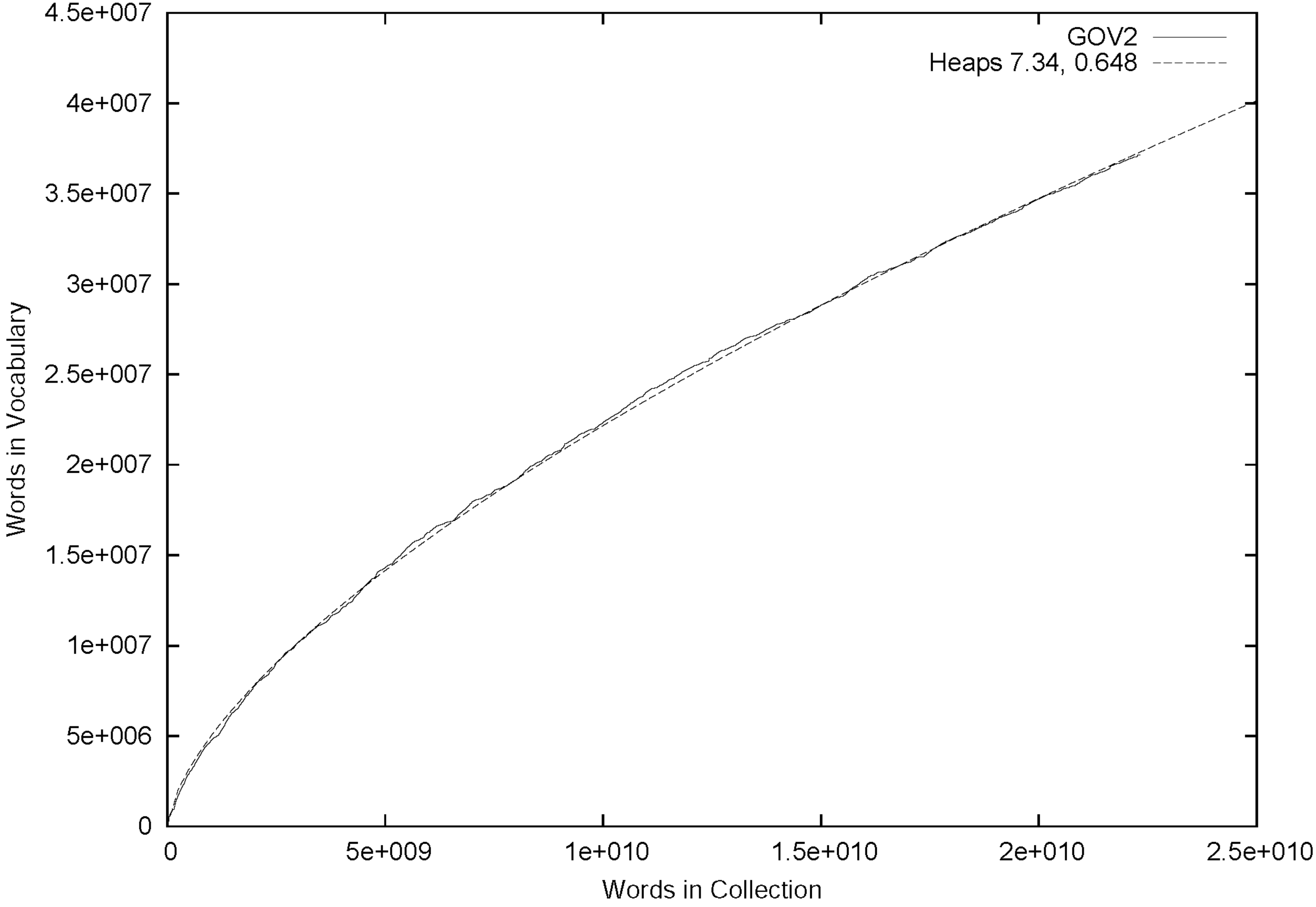
AP89 Example



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

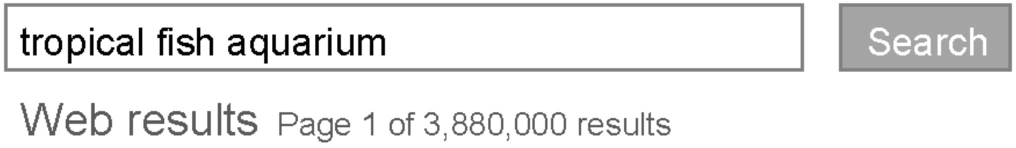
GOV2 (Web) Example



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



tropical fish aquarium

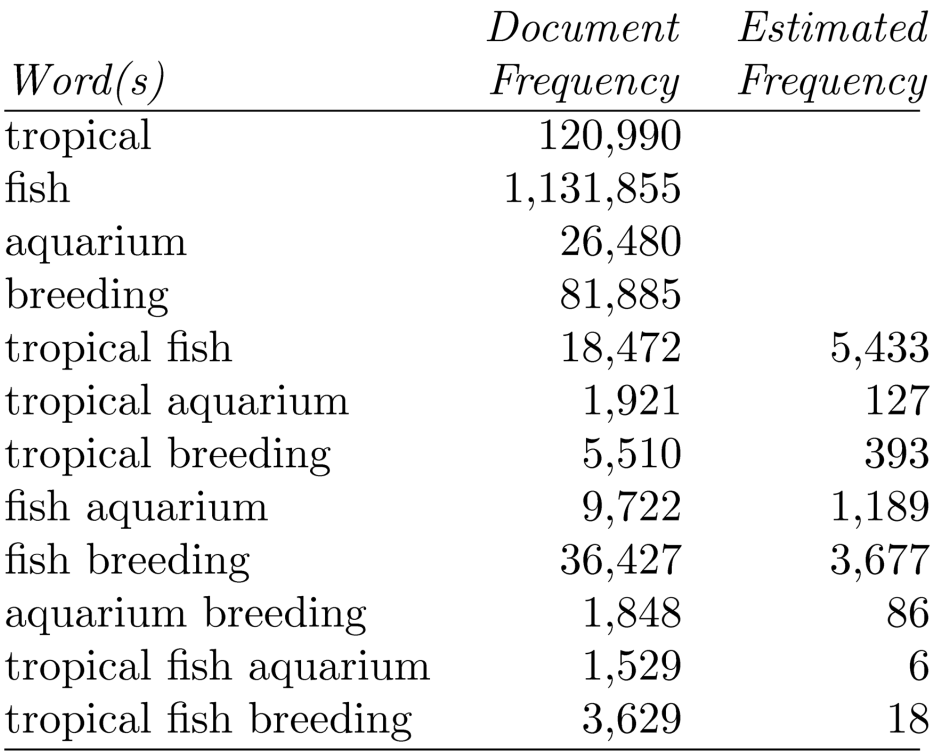
Web results Page 1 of 3,880,000 results

- 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
  - f_{abc} is the estimated size of the result set
  - f_a, f_b, f_c are the number of documents that terms *a*, *b*, and *c* occur in
  - *N* is the number of documents in the collection

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

$$\mathrm{P}(a \cap b \cap c) = \mathrm{P}(a \cap b) \cdot \mathrm{P}(c|(a \cap b))$$

$$f_{tropical \cap fish \cap aquarium} = f_{tropical \cap aquarium} \cdot f_{fish \cap aquarium} / f_{aquarium}$$

$$= 1921 \cdot 9722/26480 = 705$$

$$f_{tropical \cap fish \cap breeding} = f_{tropical \cap breeding} \cdot f_{fish \cap breeeding} / f_{breeding}$$

$$= 5510 \cdot 36427/81885 = 2451$$

Result Set Estimation

- Even better estimates using initial result set
  - Estimate is simply *C*/*s*
    - where *s* is the proportion of the total documents that have been ranked, and *C* is the number of documents found that contain all the query words
  - E.g., “tropical fish aquarium” in GOV2
    - after processing 3,000 out of the 26,480 documents that contain “aquarium”, *C* = 258

    $f_{tropical \cap fish \cap aquarium}$ = 258/(3000÷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

- Useful for auditing 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 · 771326)/3018 = 30,922,045

    (actual number is 25,205,179)

Tokenizing

- We’ve talked a lot about word counts
- But what *is* a word?
  - In English, Arabic, Chinese, Turkish, Mohawk, …
- Tokens delimited by spaces?
  - Not all languages use spaces
  - Change over time: *base ball, base-ball, baseball*
  - Spaces inside words: *ice cream, New York*
- Adopt a practical approach for now
  - Mapping strings to tokens is dimensionality reduction
    - From Σ^* to 2^Σ

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%.” becomes
  - “bigcorp 2007 annual report showed profits rose”
- Too simple for search applications or even large-scale experiments
- Why? 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
  - Trump, Apple
- Apostrophes can be a part of a word, a part of a possessive, or just a mistake
  - sandra day o’connor, 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., ccs.neu.edu, 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 to reduce dependence on query transformation components

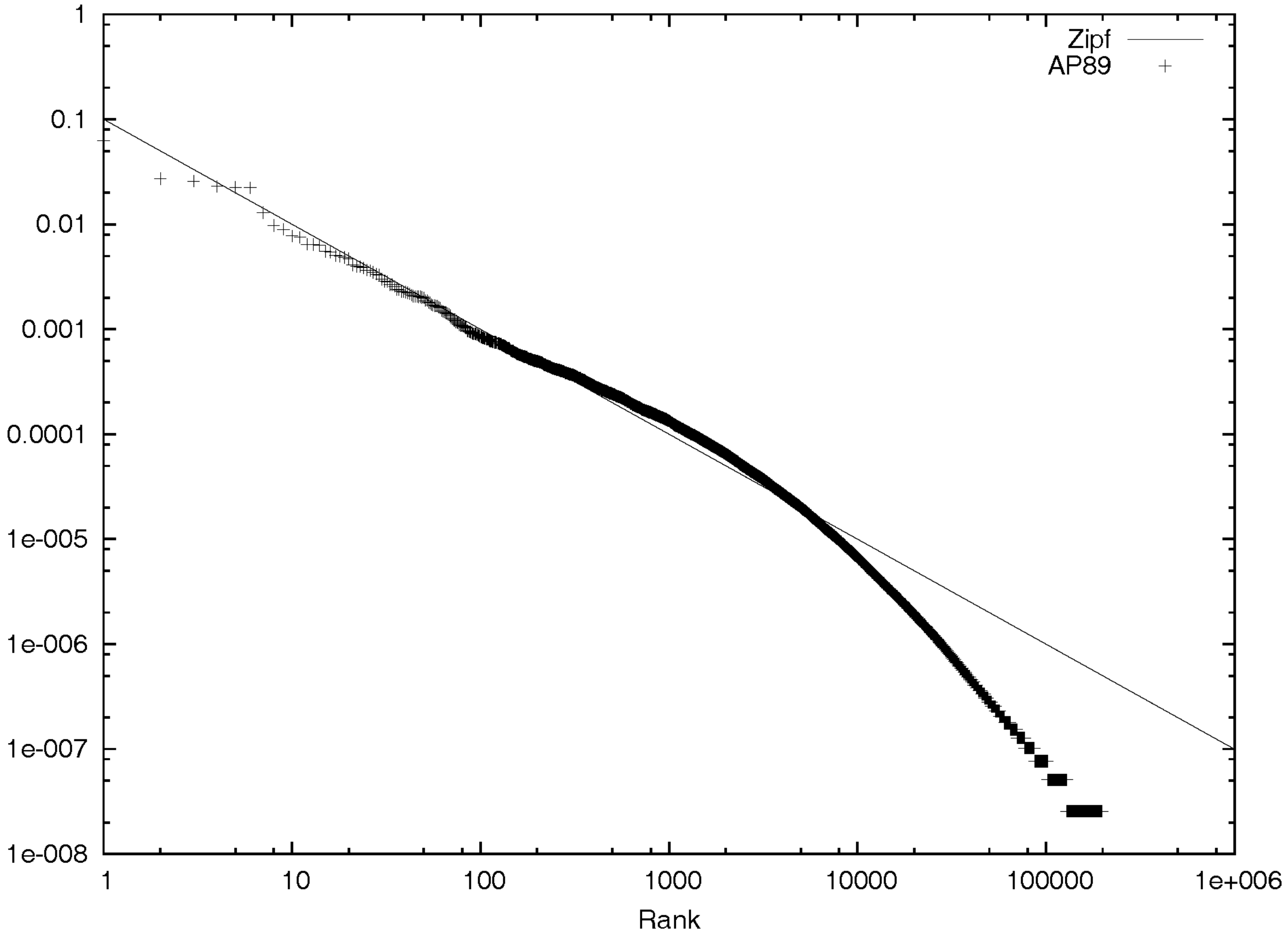
Tokenizing Process

- Not that different than simple tokenizing process used in past
- Examples of rules used with TREC
  - Apostrophes in words ignored
    - o’connor → oconnor bob’s → bobs
  - Periods in abbreviations ignored
    - I.B.M. → ibm Ph.D. → ph d
- **The future:** learned cross-language tokenizers
  - E.g., Byte-pair encoding, “word pieces”
  - Reduces vocab. size: recall/precision tradeoff

Stopping

- Function words (determiners, prepositions) have little meaning on their own
- 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”

Zipf’s Law for AP89



- Log-log plot: Note problems at high and low frequencies

Stopping

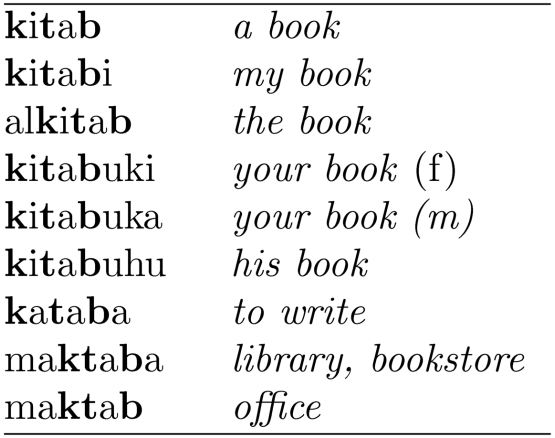
- Stopword list can be created from high-frequency words or based on a standard list
- Lists are customized for applications, domains, and even parts of documents
  - e.g., “click” is a good stopword for anchor text
- Best policy is to index all words in documents, make decisions about which words to 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 (but cf. “building”)
- Stemmers attempt to reduce morphological variations of words to a common stem
  - morphology: many-many; stemming: many-one
  - 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



ki**t**a**b**	*a book*
ki**t**a**b**i	*my book*
al**k**i**t**a**b**	*the book*
ki**t**a**b**uki	*your book* (f)
ki**t**a**b**uka	*your book (m)*
ki**t**a**b**uhu	*his book*
ka**t**a**b**a	*to write*
ma**kt**a**b**a	*library, bookstore*
ma**kt**a**b**	*office*

Words with the Arabic root **ktb**

Stemming

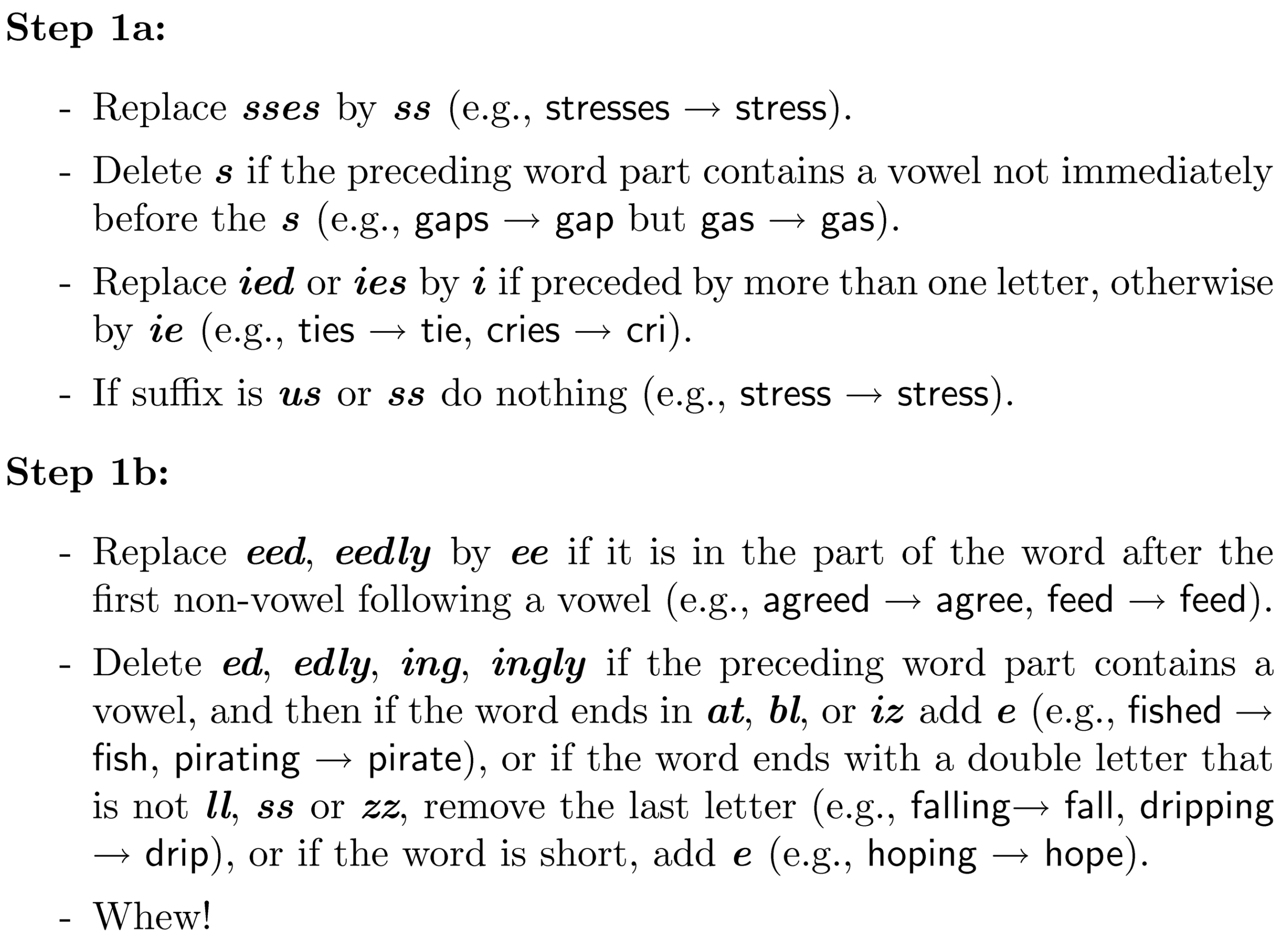
- Two basic types
  - Dictionary-based: uses lists of related words
  - Algorithmic: uses program to determine related words
- Algorithmic stemmers
  - *suffix-s:* remove ‘s’ endings assuming plural
    - e.g., cats → cat, lakes → lake, wiis → wii
    - Many *false negatives*: supplies → supplie
    - 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

Porter Stemmer

- Example step (1 of 5)



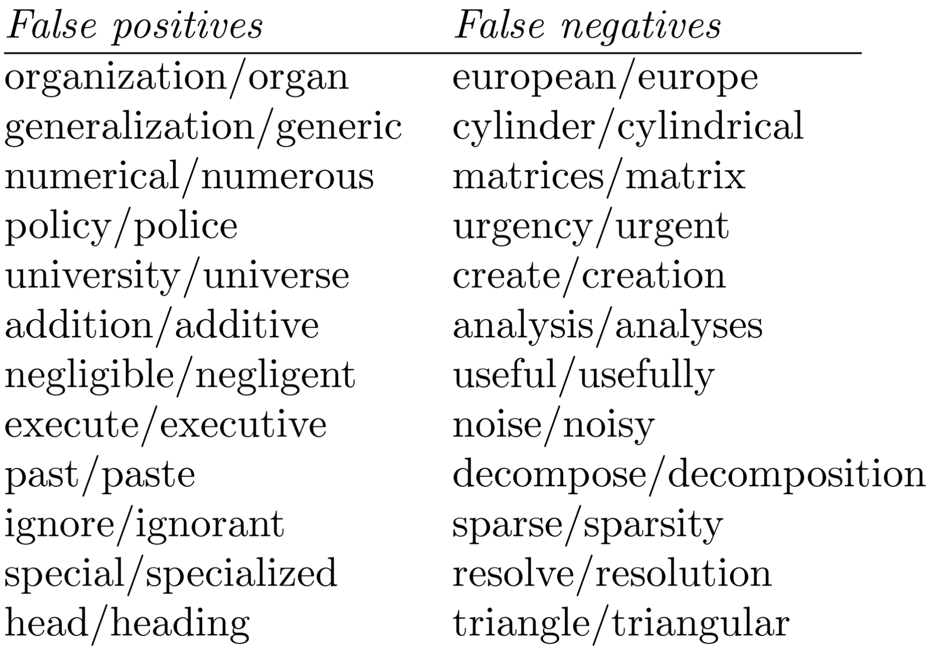
Step 1a:

- 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!

Porter Stemmer



False positives	*False negatives*
organization/organ	european/europe
generalization/generic	cylinder/cylindrical
numerical/numerous	matrices/matrix
policy/police	urgency/urgent
university/universe	create/creation
addition/additive	analysis/analyses
negligible/negligent	useful/usefully
execute/executive	noise/noisy
past/paste	decompose/decomposition
ignore/ignorant	sparse/sparsity
special/specialized	resolve/resolution
head/heading	triangle/triangular

- Porter2 stemmer addresses some of these issues
- Approach has been used with other languages

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?

Phrases

- Text processing issue – how are phrases recognized?
- Interaction w/tokenization (e.g., Chinese)
- Three possible approaches:
  - 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

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

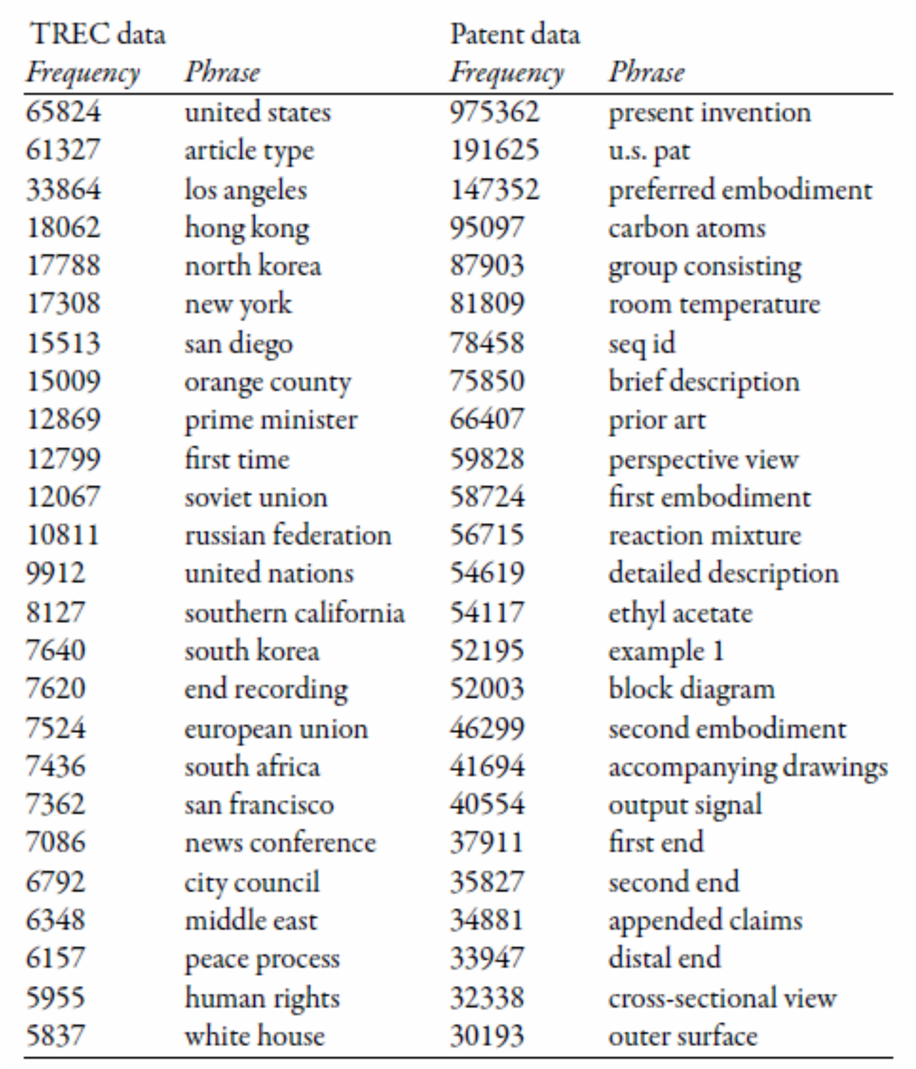
Original text:

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

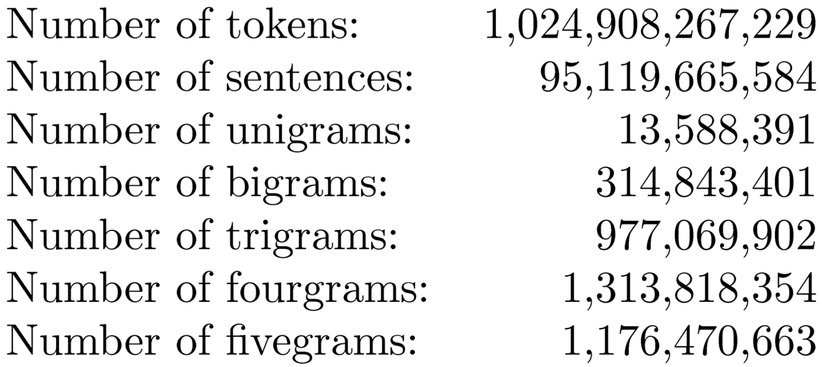
- POS tagging can be slow for large collections
- Simpler definition – phrase is any sequence of *n* words – known as *n-grams*
  - *bigram*: 2 word sequence, *trigram*: 3 word sequence, *unigram*: single words
  - N-grams also used at character level for applications such as OCR
- N-grams typically 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 $2 \le n \le 5$

Google N-Grams

- Web search engines index n-grams
- Google sample (frequency > 40):



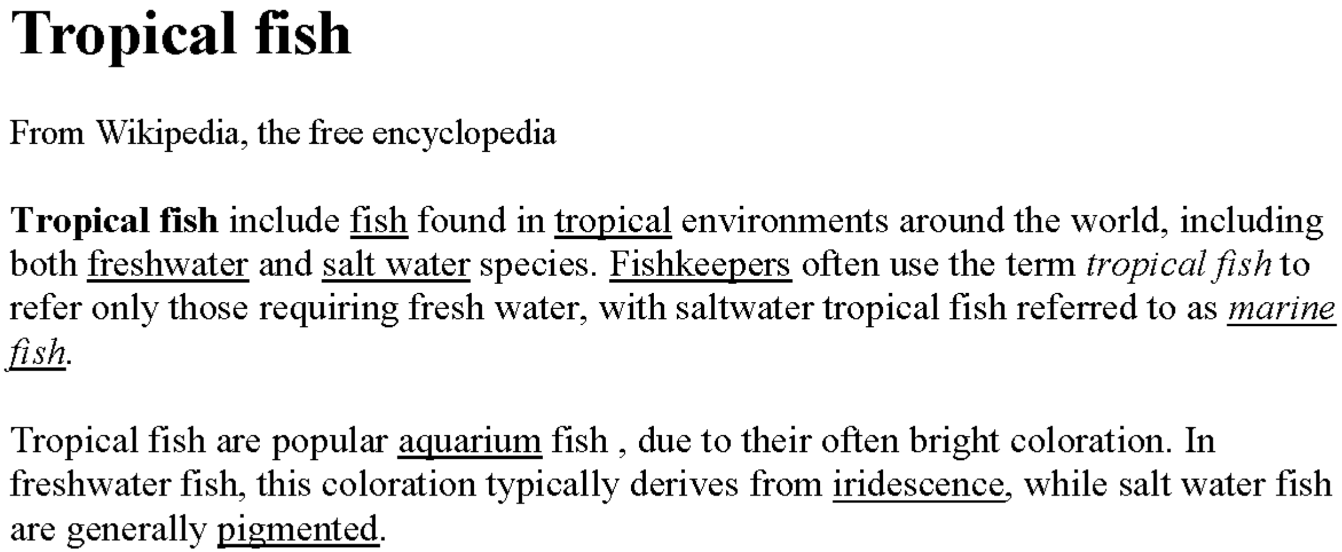
Number of tokens:	1,024,908,267,229
Number of sentences:	95,119,665,584
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663

- Most frequent trigram in English is “all rights reserved”
  - In Chinese, “limited liability corporation”

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 tropical environments around the world, including both freshwater and salt water species. Fishkeepers often use the term *tropical fish* to refer only those requiring fresh water, with saltwater tropical fish referred to as *marine fish*.

Tropical fish are popular aquarium fish , due to their often bright coloration. In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Example Web Page

```
<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., <a href="http://example.com" >Example website</a>
  - “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

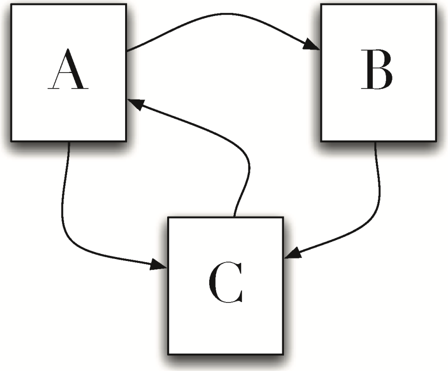
Random Surfer Model

- 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:
  - first iteration: *PR*(C) = 0.33/2 + 0.33 = 0.5, *PR*(A) = 0.33/1 = 0.33, and *PR*(B) = 0.33/2 = 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 $r < \lambda$
- $PR(C) = \lambda/3 + (1 - \lambda) \cdot (PR(A)/2 + PR(B)/1)$
- 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

```
procedure PageRank($G$)
        ▷ $G$ is the web graph, consisting of vertices (pages) and edges (links).
    $(P, L) \leftarrow G$                              ▷ Split graph into pages and links
    $I \leftarrow$ a vector of length $|P|$            ▷ The current PageRank estimate
    $R \leftarrow$ a vector of length $|P|$    ▷ The resulting better PageRank estimate
    for all entries $I_i \in I$ do
        $I_i \leftarrow 1/|P|$                 ▷ Start with each page being equally likely
    end for
    while $R$ has not converged do
        for all entries $R_i \in R$ do
            $R_i \leftarrow \lambda/|P|$ ▷ Each page has a $\lambda/|P|$ chance of random selection
        end for
        for all pages $p \in P$ do
            $Q \leftarrow$ the set of pages such that $(p, q) \in L$ and $q \in P$
            if $|Q| > 0$ then
                for all pages $q \in Q$ do
                    $R_q \leftarrow R_q + (1 - \lambda)I_p/|Q|$       ▷ Probability $I_p$ of being at page $p$
                end for
            else
                for all pages $q \in P$ do
                    $R_q \leftarrow R_q + (1 - \lambda)I_p/|P|$
                end for
            end if
            $I \leftarrow R$                   ▷ Update our current PageRank estimate
        end for
    end while
    return $R$
end procedure
```

Swap these two lines:
assignment should be outside the for loop.

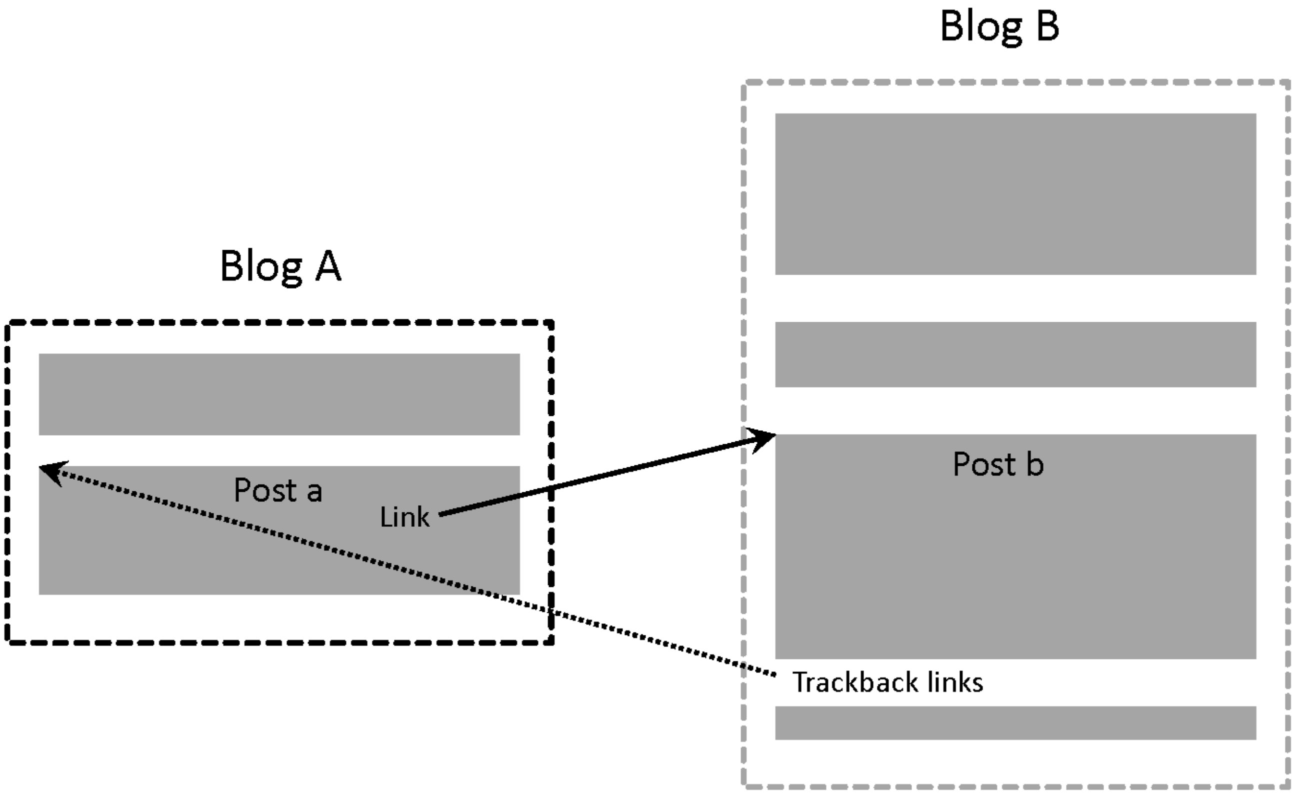
A PageRank Implementation

- Convergence check
  - Stopping criteria for this types of PR algorithm typically is of the form ||new - old|| < tau where new and old are the new and old PageRank vectors, respectively.
  - Tau is set depending on how much precision you need. Reasonable values include 0.1 or 0.01. If you want really fast, but inaccurate convergence, then you can use something like tau=1.
  - The setting of tau also depends on N (= number of documents in the collection), since ||new-old|| (for a fixed numerical precision) increases as N increases, so you can alternatively formulate your convergence criteria as ||new – old|| / N < tau.
  - Either the L1 or L2 norm can be used.

Link Quality

- Link quality is affected by spam and other factors
  - e.g., *link farms* to increase PageRank
  - *trackback links* in blogs can create loops
  - links from comments section of popular blogs
    - Blog services modify comment links to contain `rel=nofollow` attribute
    - e.g., “Come visit my <a `rel=nofollow` href="http://www.page.com">web page</a>.”

Trackback Links



PageRank in Detail

with slides from
Hinrich Schütze and Christina Lioma

Exercise: Assumptions of Link Analysis

- Assumption 1: A link on the web is a quality signal – the author of the link thinks that the linked-to page is high-quality.
- Assumption 2: The anchor text describes the content of the linked-to page.
- Is assumption 1 true in general?
- Is assumption 2 true in general?

Google bombs

Google bombs

- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.

Google bombs

- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.
- Google introduced a new weighting function in January 2007

  that fixed many Google bombs.

Google bombs

- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.
- Google introduced a new weighting function in January 2007

  that fixed many Google bombs.
- Still some remnants: [dangerous cult] on Google, Bing, Yahoo

Google bombs

- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.
- Google introduced a new weighting function in January 2007

  that fixed many Google bombs.
- Still some remnants: [dangerous cult] on Google, Bing, Yahoo
  - Coordinated link creation by those who dislike the Church of Scientology

Google bombs

- A Google bomb is a search with “bad” results due to maliciously manipulated anchor text.
- Google introduced a new weighting function in January 2007

  that fixed many Google bombs.
- Still some remnants: [dangerous cult] on Google, Bing, Yahoo
  - Coordinated link creation by those who dislike the Church of Scientology
- Defused Google bombs: [dumb motherf…], [who is a failure?], [evil empire]

Origins of PageRank: Citation analysis (1)

- Citation analysis: analysis of citations in the scientific literature.
- Example citation: "Miller (2001) has shown that physical activity alters the metabolism of estrogens."
- We can view "Miller (2001)" as a hyperlink linking two scientific articles.
- One application of these "hyperlinks" in the scientific literature:
  - Measure the similarity of two articles by the overlap of other articles citing them.
  - This is called cocitation similarity.
  - Cocitation similarity on the web: Google's "find pages like this" or "Similar" feature.

Origins of PageRank: Citation analysis (2)

- Another application: Citation frequency can be used to measure the impact of an article .
    - Simplest measure: Each article gets one vote – not very accurate.
- On the web: citation frequency = inlink count
    - A high inlink count does not necessarily mean high quality ...
    - ... mainly because of link spam.
- Better measure: weighted citation frequency or citation rank
    - An article's vote is weighted according to its citation impact.
    - Circular? No: can be formalized in a well-defined way.

Origins of PageRank: Citation analysis (3)

- Better measure: weighted citation frequency or citation rank.
- This is basically PageRank.
- PageRank was invented in the context of citation analysis by Pinsker and Narin in the 1960s.
- Citation analysis is a big deal: The budget and salary of this lecturer are / will be partly determined by the impact of his publications!

Origins of PageRank: Summary

- We can use the same formal representation for
  - citations in the scientific literature
  - hyperlinks on the web
- Appropriately weighted citation frequency is an excellent measure of quality ...
  - ... both for web pages and for scientific publications.
- Next: PageRank algorithm for computing weighted citation frequency on the web.

Model behind PageRank: Random walk

- Imagine a web surfer doing a random walk on the web
  - Start at a random page
  - At each step, go out of the current page along one of the links on that page, equiprobably
- In the steady state, each page has a long-term visit rate.
- This long-term visit rate is the page's PageRank.
- PageRank = long-term visit rate = steady state probability.

Formalization of random walk: Markov chains

Formalization of random walk: Markov chains

- A Markov chain consists of *N* states, plus an $N \times N$ transition probability matrix *P*.

Formalization of random walk: Markov chains

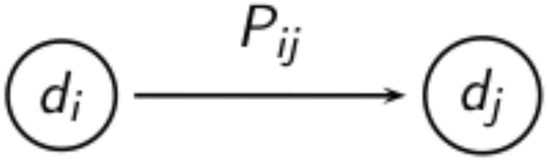
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P.
- state = page

Formalization of random walk: Markov chains

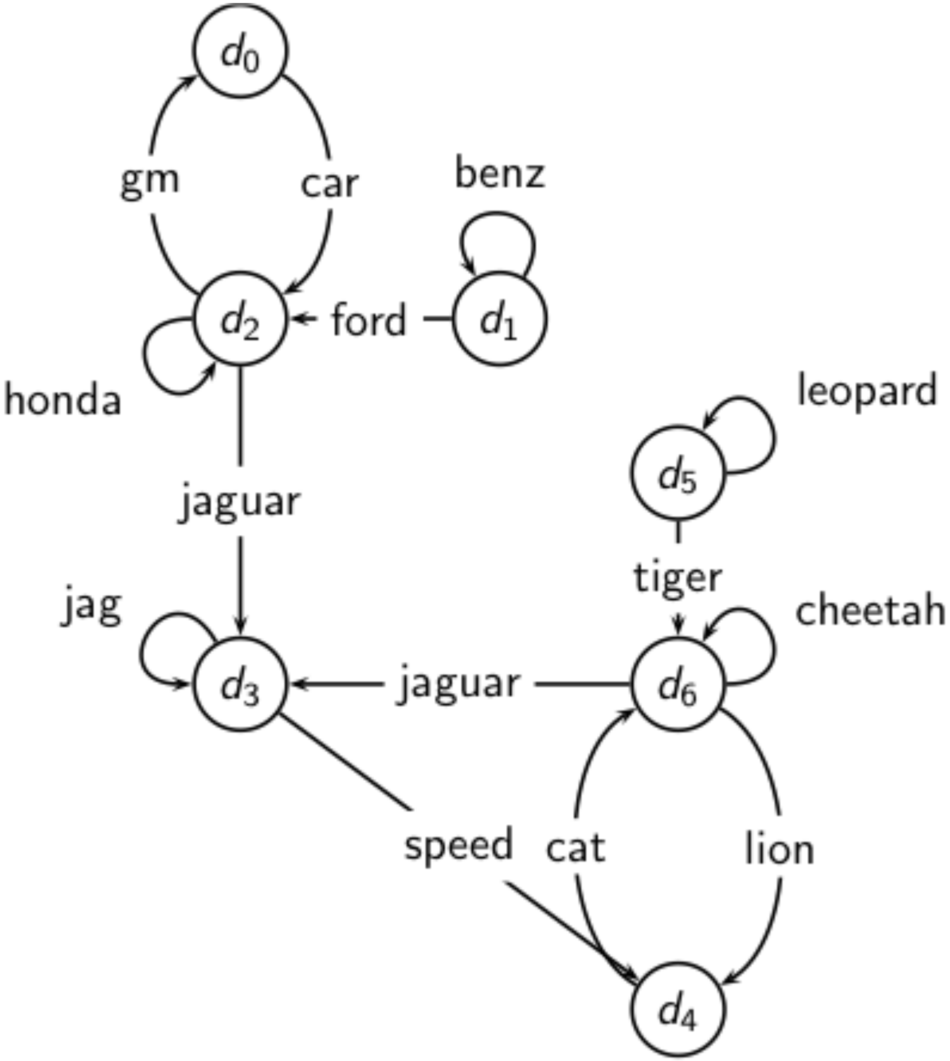
- A Markov chain consists of N states, plus an $N \times N$ transition probability matrix P.
- state = page
- At each step, we are on exactly one of the pages.

Formalization of random walk: Markov chains

- A Markov chain consists of *N* states, plus an $N \times N$ transition probability matrix *P*.
- state = page
- At each step, we are on exactly one of the pages.
- For $1 \leq i, j \leq N$, the matrix entry P_{ij} tells us the probability of *j* being the next page, given we are currently on page *i*.
- Clearly, for all i, $\sum_{j=1}^{N} P_{ij} = 1$



Example web graph



Link matrix for example

Link matrix for example

$$\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
d_1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
d_2 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\
d_3 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
d_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
d_5 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
d_6 & 0 & 0 & 0 & 1 & 1 & 0 & 1
\end{array}$$

Transition probability matrix P for example

Transition probability matrix *P* for example

$$
\begin{array}{cccccccc}
 & d_0 & d_1 & d_2 & d_3 & d_4 & d_5 & d_6 \\
d_0 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
d_1 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 & 0.00 & 0.00 \\
d_2 & 0.33 & 0.00 & 0.33 & 0.33 & 0.00 & 0.00 & 0.00 \\
d_3 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 & 0.00 & 0.00 \\
d_4 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \\
d_5 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.50 & 0.50 \\
d_6 & 0.00 & 0.00 & 0.00 & 0.33 & 0.33 & 0.00 & 0.33
\end{array}
$$

Long-term visit rate

Long-term visit rate

- Recall: PageRank = long-term visit rate.

Long-term visit rate

- Recall: PageRank = long-term visit rate.
- Long-term visit rate of page *d* is the probability that a web surfer is at page *d* at a given point in time.

Long-term visit rate

- Recall: PageRank = long-term visit rate.
- Long-term visit rate of page *d* is the probability that a web surfer is at page *d* at a given point in time.
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?

Long-term visit rate

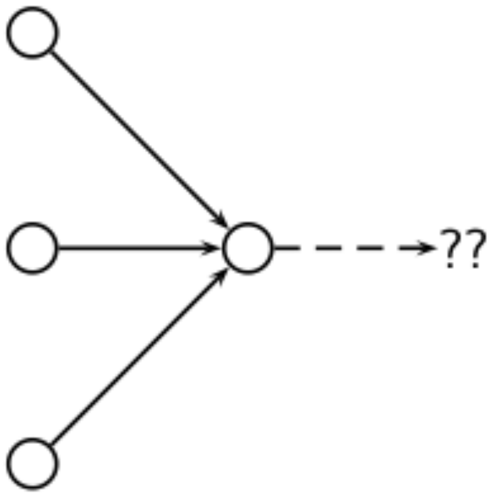
- Recall: PageRank = long-term visit rate.
- Long-term visit rate of page *d* is the probability that a web surfer is at page *d* at a given point in time.
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?
- The web graph must correspond to an ergodic Markov chain.

Long-term visit rate

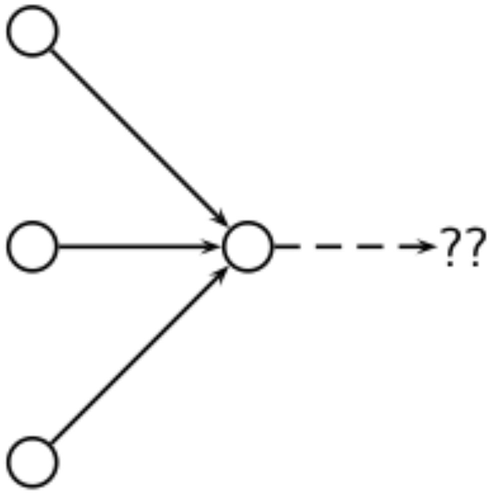
- Recall: PageRank = long-term visit rate.
- Long-term visit rate of page *d* is the probability that a web surfer is at page *d* at a given point in time.
- Next: what properties must hold of the web graph for the long-term visit rate to be well defined?
- The web graph must correspond to an ergodic Markov chain.
- First a special case: The web graph must not contain dead ends.

Dead ends

Dead ends

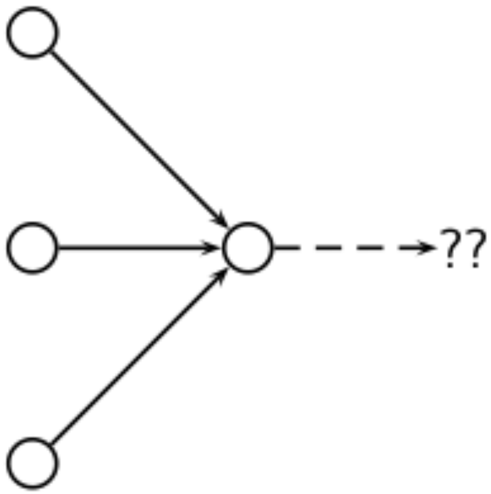


Dead ends



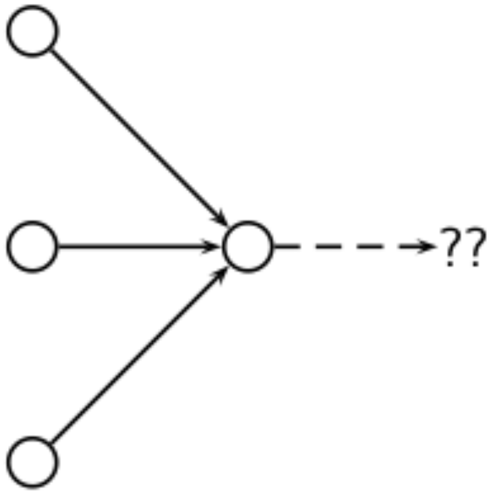
- The web is full of dead ends.

Dead ends



- The web is full of dead ends.
- Random walk can get stuck in dead ends.

Dead ends



- The web is full of dead ends.
- Random walk can get stuck in dead ends.
- If there are dead ends, long-term visit rates are not well-defined.

Teleporting – to get us of dead ends

Teleporting – to get us of dead ends

- At a dead end, jump to a random web page with prob. 1/ *N* .

Teleporting – to get us of dead ends

- At a dead end, jump to a random web page with prob. 1/ *N* .
- At a non-dead end, with probability 10%, jump to a random web page (to each with a probability of 0.1/*N*).

Teleporting – to get us of dead ends

- At a dead end, jump to a random web page with prob. 1/ *N* .
- At a non-dead end, with probability 10%, jump to a random web page (to each with a probability of 0.1/*N*).
- With remaining probability (90%), go out on a random hyperlink.

Teleporting – to get us of dead ends

- At a dead end, jump to a random web page with prob. $1/N$.
- At a non-dead end, with probability 10%, jump to a random web page (to each with a probability of $0.1/N$).
- With remaining probability (90%), go out on a random hyperlink.
  - For example, if the page has 4 outgoing links: randomly choose one with probability (1-0.10)/4=0.225

Teleporting – to get us of dead ends

- At a dead end, jump to a random web page with prob. 1/ *N* .
- At a non-dead end, with probability 10%, jump to a random web page (to each with a probability of 0.1/*N*).
- With remaining probability (90%), go out on a random hyperlink.
  - For example, if the page has 4 outgoing links: randomly choose one with probability (1-0.10)/4=0.225
- 10% is a parameter, the teleportation rate.

Teleporting – to get us of dead ends

- At a dead end, jump to a random web page with prob. 1/ *N* .
- At a non-dead end, with probability 10%, jump to a random web page (to each with a probability of 0.1/*N*).
- With remaining probability (90%), go out on a random hyperlink.
    - For example, if the page has 4 outgoing links: randomly choose one with probability (1-0.10)/4=0.225
- 10% is a parameter, the teleportation rate.
- Note: “jumping” from dead end is independent of teleportation rate.

Result of teleporting

Result of teleporting

- With teleporting, we cannot get stuck in a dead end.

Result of teleporting

- With teleporting, we cannot get stuck in a dead end.
- But even without dead ends, a graph may not have well-defined long-term visit rates.

Result of teleporting

- With teleporting, we cannot get stuck in a dead end.
- But even without dead ends in the original graph, we may not have well-defined long-term visit rates.
- More generally, we require that the Markov chain be ergodic.

Ergodic Markov chains

- A Markov chain is ergodic if it is irreducible and aperiodic.

Ergodic Markov chains

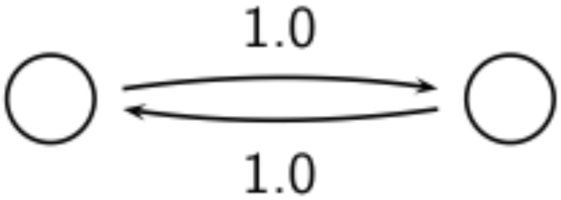
- A Markov chain is ergodic if it is irreducible and aperiodic.
- Irreducibility. Roughly: there is a path from any other page.

Ergodic Markov chains

- A Markov chain is ergodic if it is irreducible and aperiodic.
- Irreducibility. Roughly: there is a path from any other page.
- Aperiodicity. Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially.

Ergodic Markov chains

- A Markov chain is ergodic if it is irreducible and aperiodic.
- Irreducibility. Roughly: there is a path from any other page.
- Aperiodicity. Roughly: The pages cannot be partitioned such that the random walker visits the partitions sequentially.
- A non-ergodic Markov chain:



Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.
- It doesn’t matter where we start.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.
- It doesn’t matter where we start.
- Teleporting makes the web graph ergodic.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.
- It doesn't matter where we start.
- Teleporting makes the web graph ergodic.
- $\Longrightarrow$Web-graph+teleporting has a steady-state probability distribution.

Ergodic Markov chains

- Theorem: For any ergodic Markov chain, there is a unique long-term visit rate for each state.
- This is the steady-state probability distribution.
- Over a long time period, we visit each state in proportion to this rate.
- It doesn't matter where we start.
- Teleporting makes the web graph ergodic.
- $\Longrightarrow$Web-graph+teleporting has a steady-state probability distribution.
- $\Longrightarrow$Each page in the web-graph+teleporting has a PageRank.

Formalization of “visit”: Probability vector

Formalization of “visit”: Probability vector

- A probability (row) vector $\vec{x} = (x_1 , ..., x_N)$ tells us where the random walk is at any point.

Formalization of “visit”: Probability vector

- A probability (row) vector $\vec{x} = (x_1 , ..., x_N)$ tells us where the random walk is at any point.
- Example:

(	0	0	0	…	1	…	0	0	0	)
	1	2	3	…	*i*	…	N-2	N-1	N	

Formalization of “visit”: Probability vector

- A probability (row) vector $\vec{x} = (x_1 , ..., x_N)$ tells us where the random walk is at any point.
- Example:

(	0	0	0	…	1	…	0	0	0	)
	1	2	3	…	*i*	…	N-2	N-1	N	

- More generally: the random walk is on the page *i* with probability x_i.

Formalization of “visit”: Probability vector

- A probability (row) vector $\vec{x} = (x_1 , ..., x_N)$ tells us where the random walk is at any point.
- Example:

(	0	0	0	…	1	…	0	0	0	)
	1	2	3	…	*i*	…	N-2	N-1	N	

- More generally: the random walk is on the page *i* with probability x_i.
- Example:

(	0.05	0.01	0.0	…	0.2	…	0.01	0.05	0.03	)
	1	2	3	…	*i*	…	N-2	N-1	N	

Formalization of “visit”: Probability vector

- A probability (row) vector $\vec{x} = (x_1 , ..., x_N)$ tells us where the random walk is at any point.
- Example:

(	0	0	0	…	1	…	0	0	0	)
	1	2	3	…	*i*	…	N-2	N-1	N	

- More generally: the random walk is on the page *i* with probability x_i.
- Example:

(	0.05	0.01	0.0	…	0.2	…	0.01	0.05	0.03	)
	1	2	3	…	*i*	…	N-2	N-1	N	

- $\Sigma\, x_i = 1$

Change in probability vector

- If the probability vector is $\vec{x} = (x_1 , ..., x_N)$, at this step, what is it at the next step?

Change in probability vector

- If the probability vector is $\vec{x} = (x_1 , ..., x_N)$, at this step, what is it at the next step?
- Recall that row *i* of the transition probability matrix *P* tells us where we go next from state *i.*

Change in probability vector

- If the probability vector is $\vec{x} = (x_1 , ..., x_N),$ at this step, what is it at the next step?
- Recall that row i of the transition probability matrix P tells us where we go next from state $i.$
- So from $\vec{x}$, our next state is distributed as $\vec{x}P.$

Steady state in vector notation

Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ of probabilities.

Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ of probabilities.
- (We use $\vec{\pi}$ to distinguish it from the notation for the probability vector $\vec{x}$.)

Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ of probabilities.
- (We use $\vec{\pi}$ to distinguish it from the notation for the probability vector $\vec{x}$.)
- π is the long-term visit rate (or PageRank) of page *i*.

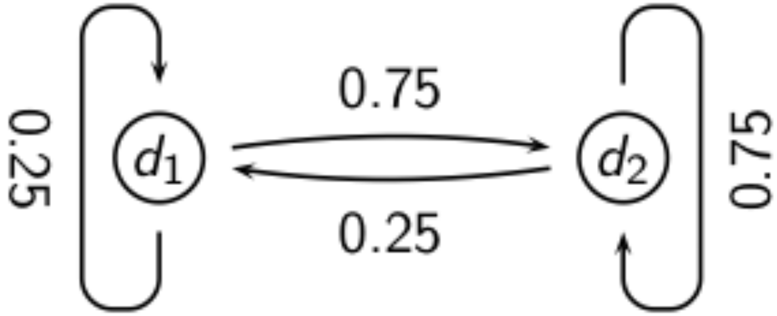
Steady state in vector notation

- The steady state in vector notation is simply a vector $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ of probabilities.
- (We use $\vec{\pi}$ to distinguish it from the notation for the probability vector $\vec{x}$.)
- π is the long-term visit rate (or PageRank) of page *i*.
- So we can think of PageRank as a very long vector – one entry per page.

Steady-state distribution: Example

Steady-state distribution: Example

- What is the PageRank / steady state in this example?



Steady-state distribution: Example

Steady-state distribution: Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.25$ $P_{21} = 0.25$	$P_{12} = 0.75$ $P_{22} = 0.75$
t_0	0.25	0.75		
t_1				

PageRank vector = $\pi = (\pi_1, \pi_2) = (0.25, 0.75)$

$P_\mathrm{t}(d_1) = P_{\mathrm{t-1}}(d_1) * P_{11} + P_{\mathrm{t-1}}(d_2) * P_{21}$
$P_\mathrm{t}(d_2) = P_{\mathrm{t-1}}(d_1) * P_{12} + P_{\mathrm{t-1}}(d_2) * P_{22}$

Steady-state distribution: Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.25$ $P_{21} = 0.25$	$P_{12} = 0.75$ $P_{22} = 0.75$
t_0	0.25	0.75	0.25	0.75
t_1				

PageRank vector = $\pi = (\pi_1\overset{\rightarrow}{,}\pi_2) = (0.25, 0.75)$

$P_t(d_1) = P_{t\text{-}1}(d_1) * P_{11} + P_{t\text{-}1}(d_2) * P_{21}$
$P_t(d_2) = P_{t\text{-}1}(d_1) * P_{12} + P_{t\text{-}1}(d_2) * P_{22}$

Steady-state distribution: Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.25$ $P_{21} = 0.25$	$P_{12} = 0.75$ $P_{22} = 0.75$
t_0	0.25	0.75	0.25	0.75
t_1	0.25	0.75		

PageRank vector = $\pi = (\pi_1 \vec{,} \pi_2)$ = (0.25, 0.75)

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$
$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Steady-state distribution: Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.25$ $P_{21} = 0.25$	$P_{12} = 0.75$ $P_{22} = 0.75$
t_0	0.25	0.75	0.25	0.75
t_1	0.25	0.75	(convergence)	

PageRank vector = $\pi = (\pi_1, \pi_2)$ = (0.25, 0.75)

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

How do we compute the steady state vector?

- In other words: how do we compute PageRank?

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi} P$

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- ... and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi} P$
- Solving this matrix equation gives us $\vec{\pi}$.

Computational complexity?

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi} P$
- Solving this matrix equation gives us $\vec{\pi}$.
- $\vec{\pi}$ is the principal left eigenvector for P …

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi} P$
- Solving this matrix equation gives us $\vec{\pi}$.
- $\vec{\pi}$ is the principal left eigenvector for P …
- … that is, $\vec{\pi}$ is the left eigenvector with the largest eigenvalue.

How do we compute the steady state vector?

- In other words: how do we compute PageRank?
- Recall: $\vec{\pi} = (\pi_1, \pi_2, \ldots, \pi_N)$ is the PageRank vector, the vector of steady-state probabilities ...
- … and if the distribution in this step is $\vec{x}$, then the distribution in the next step is $\vec{x}P$.
- But $\vec{\pi}$ is the steady state!
- So: $\vec{\pi} = \vec{\pi} P$
- Solving this matrix equation gives us $\vec{\pi}$.
- $\vec{\pi}$ is the principal left eigenvector for P …
- … that is, $\vec{\pi}$ is the left eigenvector with the largest eigenvalue.
- All transition probability matrices have largest eigenvalue 1.

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After *k* steps, we're at $\vec{x}P^k$.

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After *k* steps, we're at $\vec{x}P^k$.
- Algorithm: multiply $\vec{x}$ by increasing powers of *P* until convergence.

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After *k* steps, we're at $\vec{x}P^k$.
- Algorithm: multiply $\vec{x}$ by increasing powers of *P* until convergence.
- This is called the power method.

One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After k steps, we're at $\vec{x}P^k$.
- Algorithm: multiply $\vec{x}$ by increasing powers of P until convergence.
- This is called the power method.
- Recall: regardless of where we start, we eventually reach the steady state $\vec{\pi}$.

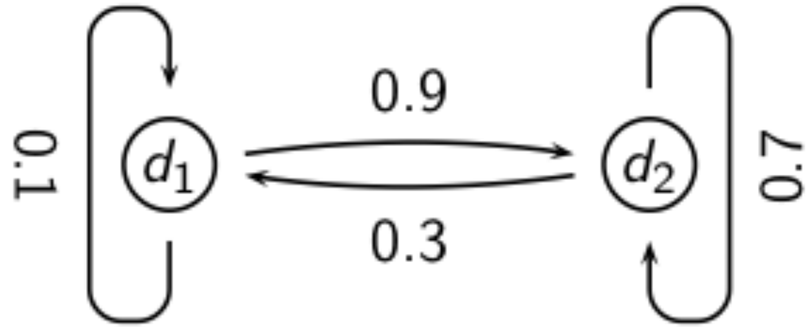
One way of computing the PageRank $\vec{\pi}$

- Start with any distribution $\vec{x}$, e.g., uniform distribution
- After one step, we're at $\vec{x}P$.
- After two steps, we're at $\vec{x}P^2$.
- After *k* steps, we're at $\vec{x}P^k$.
- Algorithm: multiply $\vec{x}$ by increasing powers of P until convergence.
- This is called the power method.
- Recall: regardless of where we start, we eventually reach the steady state $\vec{\pi}$.
- Thus: we will eventually (asymptotically) reach the steady state.

Power method: Example

Power method: Example

- What is the PageRank / steady state in this example?



Computing PageRank: Power Example

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1			$= \vec{x}P$
t_1					$= \vec{x}P^2$
t_2					$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$
$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1					$= \vec{x}P^2$
t_2					$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$
$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7			$= \vec{x}P^2$
t_2					$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$P_t(d_1) = P_{t\text{-}1}(d_1) * P_{11} + P_{t\text{-}1}(d_2) * P_{21}$

$P_t(d_2) = P_{t\text{-}1}(d_1) * P_{12} + P_{t\text{-}1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2					$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$P_t(d_1) = P_{t\text{-}1}(d_1) * P_{11} + P_{t\text{-}1}(d_2) * P_{21}$
$P_t(d_2) = P_{t\text{-}1}(d_1) * P_{12} + P_{t\text{-}1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76			$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3					$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748			$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
					. . .
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
				. . .	. . .
t_∞					$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$$
$$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
			. . .		. . .
t_∞	0.25	0.75			$= \vec{x}P^\infty$

$$P_t(d_1) = P_{t\text{-}1}(d_1) \ast P_{11} + P_{t\text{-}1}(d_2) \ast P_{21}$$
$$P_t(d_2) = P_{t\text{-}1}(d_1) \ast P_{12} + P_{t\text{-}1}(d_2) \ast P_{22}$$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
			. . .		. . .
t_∞	0.25	0.75	0.25	0.75	$= \vec{x}P^\infty$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$
$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Computing PageRank: Power Example

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$			
			$P_{11} = 0.1$ $P_{21} = 0.3$	$P_{12} = 0.9$ $P_{22} = 0.7$	
t_0	0	1	0.3	0.7	$= \vec{x}P$
t_1	0.3	0.7	0.24	0.76	$= \vec{x}P^2$
t_2	0.24	0.76	0.252	0.748	$= \vec{x}P^3$
t_3	0.252	0.748	0.2496	0.7504	$= \vec{x}P^4$
			. . .		. . .
t_∞	0.25	0.75	0.25	0.75	$= \vec{x}P^\infty$

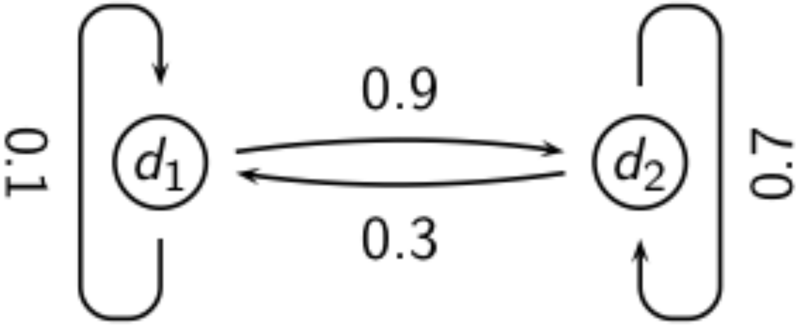
PageRank vector = $\pi = (\overset{\rightarrow}{\pi_1,}\pi_2) = (0.25, 0.75)$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Power method: Example

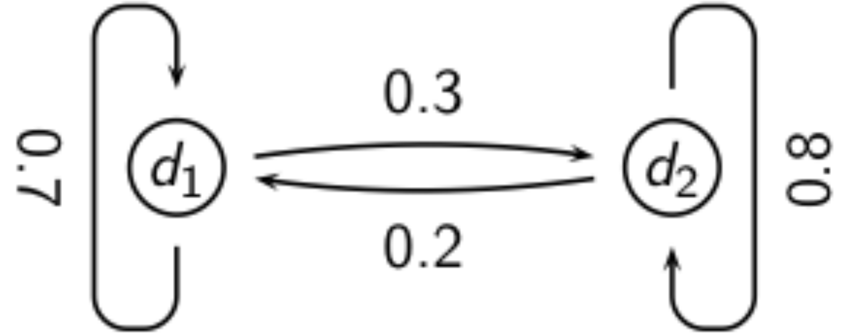
- What is the PageRank / steady state in this example?



- The steady state distribution (= the PageRanks) in this example are 0.25 for d_1 and 0.75 for d_2.

Exercise: Compute PageRank using power method

Exercise: Compute PageRank using power method



Solution

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1		
t_1				
t_2				
t_3				
t_∞				

PageRank vector = $\vec{\pi} = (\pi_1, \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1				
t_2				
t_3				
t_∞				

PageRank vector = $\vec{\pi} = (\pi_1, \pi_2) = (0.4, 0.6)$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8		
t_2				
t_3				
t_∞				

PageRank vector = $\vec{\pi} = (\pi_1, \pi_2) = (0.4, 0.6)$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2				
t_3				
t_∞				

PageRank vector = $\pi = (\pi_1\overset{\rightarrow}{,}\pi_2) = (0.4, 0.6)$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7		
t_3				
t_∞				

PageRank vector = $\vec{\pi} = (\pi_1, \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t\text{-}1}(d_1) * P_{11} + P_{t\text{-}1}(d_2) * P_{21}$

$P_t(d_2) = P_{t\text{-}1}(d_1) * P_{12} + P_{t\text{-}1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3				
t_∞				

PageRank vector = π = $(\pi_1 \overset{\rightarrow}{,} \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65		
t_∞				

PageRank vector = $\pi = (\pi_{\overset{\rightarrow}{1},} \pi_2) = (0.4, 0.6)$

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65	0.375	0.625
t_∞				

PageRank vector = $\pi = (\pi_{\vec{1}}, \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65	0.375	0.625
			. . .	
t_∞				

PageRank vector = $\vec{\pi} = (\pi_1, \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t-1}(d_1) * P_{11} + P_{t-1}(d_2) * P_{21}$

$P_t(d_2) = P_{t-1}(d_1) * P_{12} + P_{t-1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65	0.375	0.625
			. . .	
t_∞	0.4	0.6		

PageRank vector = $\pi = (\pi_{\overset{\to}{1}}, \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t\text{-}1}(d_1) * P_{11} + P_{t\text{-}1}(d_2) * P_{21}$

$P_t(d_2) = P_{t\text{-}1}(d_1) * P_{12} + P_{t\text{-}1}(d_2) * P_{22}$

Solution

	x_1 $P_t(d_1)$	x_2 $P_t(d_2)$		
			$P_{11} = 0.7$ $P_{21} = 0.2$	$P_{12} = 0.3$ $P_{22} = 0.8$
t_0	0	1	0.2	0.8
t_1	0.2	0.8	0.3	0.7
t_2	0.3	0.7	0.35	0.65
t_3	0.35	0.65	0.375	0.625
			. . .	
t_∞	0.4	0.6	0.4	0.6

PageRank vector = $\pi = (\pi_1 \overset{\rightarrow}{,} \pi_2)$ = (0.4, 0.6)

$P_t(d_1) = P_{t\text{-}1}(d_1) * P_{11} + P_{t\text{-}1}(d_2) * P_{21}$

$P_t(d_2) = P_{t\text{-}1}(d_1) * P_{12} + P_{t\text{-}1}(d_2) * P_{22}$

PageRank summary

PageRank summary

- Offline processing

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page *i*.

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page *i*.
- Query processing

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page *i*.
- Query processing
  - Retrieve pages satisfying the query

PageRank summary

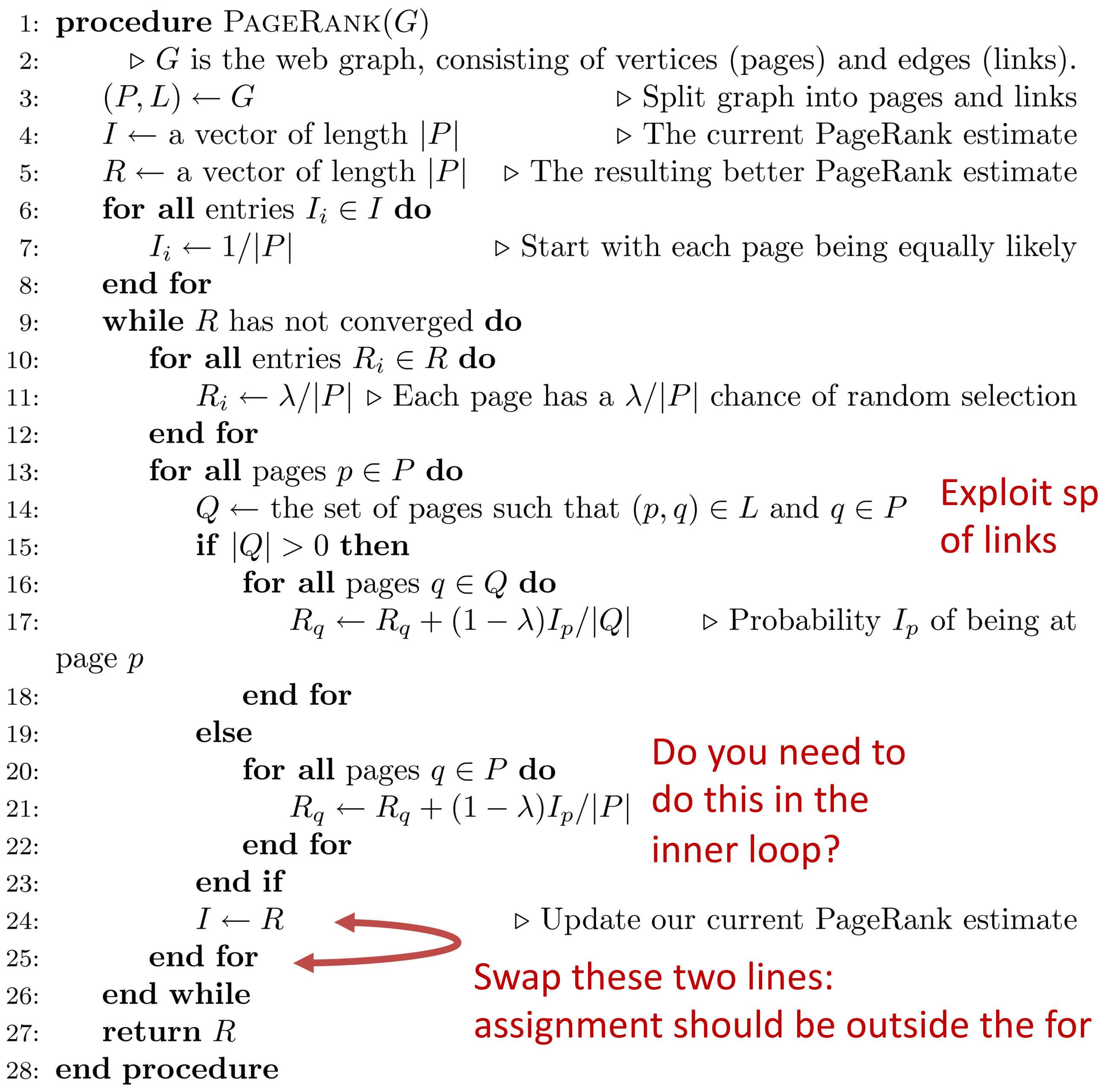
- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page *i*.
- Query processing
  - Retrieve pages satisfying the query
  - Rank them by their PageRank

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page *i*.
- Query processing
  - Retrieve pages satisfying the query
  - Rank them by their PageRank
  - Return reranked list to the user

PageRank summary

- Offline processing
  - Given graph of links, build matrix *P*
  - Apply teleportation
  - From modified matrix, compute $\vec{\pi}$
  - $\vec{\pi}_i$ is the PageRank of page *i*.
- Query processing
  - Retrieve pages satisfying the query
  - Rank them by their PageRank
  - Return reranked list to the user
- Question: How should we represent matrix *P* w/teleportation?
  - How much memory will be need?



```
procedure PageRank($G$)
        ▷ $G$ is the web graph, consisting of vertices (pages) and edges (links).
    $(P, L) \leftarrow G$                                  ▷ Split graph into pages and links
    $I \leftarrow$ a vector of length $|P|$                 ▷ The current PageRank estimate
    $R \leftarrow$ a vector of length $|P|$       ▷ The resulting better PageRank estimate
    for all entries $I_i \in I$ do
        $I_i \leftarrow 1/|P|$                    ▷ Start with each page being equally likely
    end for
    while $R$ has not converged do
        for all entries $R_i \in R$ do
            $R_i \leftarrow \lambda/|P|$ ▷ Each page has a $\lambda/|P|$ chance of random selection
        end for
        for all pages $p \in P$ do
            $Q \leftarrow$ the set of pages such that $(p, q) \in L$ and $q \in P$
            if $|Q| > 0$ then
                for all pages $q \in Q$ do
                    $R_q \leftarrow R_q + (1 - \lambda)I_p/|Q|$       ▷ Probability $I_p$ of being at page $p$
                end for
            else
                for all pages $q \in P$ do
                    $R_q \leftarrow R_q + (1 - \lambda)I_p/|P|$
                end for
            end if
            $I \leftarrow R$                         ▷ Update our current PageRank estimate
        end for
    end while
    return $R$
end procedure
```

Exploit sparsity of links

Do you need to do this in the inner loop?

Swap these two lines:
assignment should be outside the for loop.

How important is PageRank?

- Frequent claim: PageRank is the most important component of web ranking.
- The reality:
    - There are several components that are at least as important: e.g., anchor text, phrases, proximity, tiered indexes ...
    - Rumor has it that PageRank in his original form (as presented here) now has a negligible impact on ranking!
    - However, variants of a page's PageRank are still an essential part of ranking.
    - Addressing link spam is difficult and crucial.