Information Retrieval
INFO/CS 4300

- Instructor: Chris Buckley
  – Office hours Wednesdays 11am Gates 231
- Piazza will be the main communication tool
  – https://piazza.com/cornell/fall2014/info4300/home
  – Lecture notes will appear there.
  – TA office hours and locations appear there.

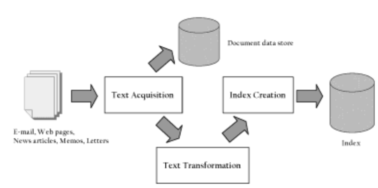
Admin & logistics

- Course enrollment is now open
- CMS is being used for assignments. See Piazza message for links to it and documentation. If you sign in to CMS and do NOT see Info 4300 as a course for you, tell us either through Piazza (private to instructors)
- First Critique due today (paper by Brin,Page)

Previous lectures:

- Lectures 1 & 2: Overview of course and Search Engines
  – Material in Croft chapters 1 and 2. Manning 1
- Lecture 3: Test collection evaluation
  – In Croft Chap 8, Manning Chap 8, and lecture notes
- Lecture 4: Start detailed Search Engines
  – Text Acquisition - Web Crawling
  – In Croft Chap 3, Manning Chap 20
- Today: Finish Text Acquisition, start Indexing.
  – Croft Chap 4, Manning Chap 2

Indexing Process

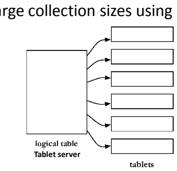


Text Acquisition and Web Crawlers

- Web crawlers
  - Retrieving web pages
  - Crawling the web
    - Desktop crawlers
    - Document feeds
  - File conversion
  - Storing the documents
  - Removing noise

BigTable

- Google's document storage system
  - http://research.google.com/archive/bigtable.html
  - Customized for storing, finding, and updating web pages
  - Handles large collection sizes using inexpensive computers

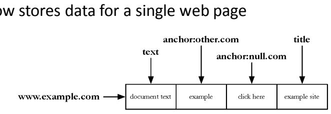


BigTable

- No query language, no complex queries to optimize
- Only row-level transactions
- Tablets are stored in a replicated file system that is accessible by all BigTable servers
- Any changes to a BigTable tablet are recorded to a transaction log, which is also stored in a shared file system
- If a tablet crashes, easy recovery from transaction logs
- If any tablet server crashes, another server can immediately read the tablet data and transaction log from the file system and take over

BigTable

- Logically organized into rows
- A row stores data for a single web page



- Combination of a row key and a column key point to a single *cell* in the row, and then a timestamp value points to the data within the cell.

BigTable

- BigTable can have a huge number of columns per row
  - all rows have the same column groups
    - Eg, content, language, anchor text
  - not all rows have the same columns
  - important for reducing disk reads to access document data
    - Or reducing memory accesses if all in memory!
- Rows are partitioned into tablets based on their row keys
  - simplifies determining which tablet is appropriate

Sample Google Tables (2006!)

Project Name	Table Size (TB)	Compression ratio	# cells (billions)	# Column Families	% in memory
Crawl	800	11%	1000	16	0
Google Analytics	200	14%	80	1	0
Google Earth	0.5	64%	7	2	33
Google Earth	70	-	9	8	0

Overview again

- Web crawlers
  - Retrieving web pages
  - Crawling the web
    - Desktop crawlers
    - Document feeds
  - File conversion
  - Storing the documents
  - Removing noise

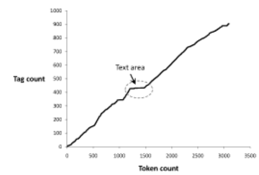
Removing Noise

- Many web pages contain text, links, and pictures that are not directly related to the main content of the page
- This additional material is mostly *noise* that could negatively affect the ranking of the page
- Techniques have been developed to detect the content blocks in a web page
  - Non-content material is either ignored or reduced in importance in the indexing process

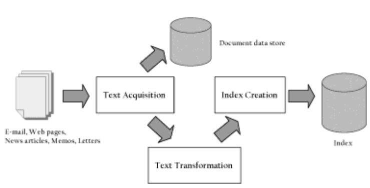
- Noise Example

Finding Content Blocks

- Cumulative distribution of tags in the example web page
  - Main text content of the page corresponds to the "plateau" in the middle of the distribution



Indexing Process



Processing Text

- Converting documents to a more consistent set of ***index terms***
- Why?
  - Sometimes not clear where words begin and end
    - Not even clear what a word is in some languages
      - e.g., Chinese, Korean
  - 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 (often)
  - Not all words are of equal value in a search

Text transformation

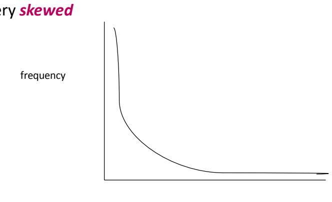
- Word occurrence statistics
- Tokenizing
- Stopping and stemming

Text Statistics

- Many statistical characteristics of word occurrences are predictable
- Retrieval models and ranking algorithms depend heavily on them
  - e.g., important words occur often in documents but are not high frequency in collection [Luhn, 1958]

Distribution of word frequencies

- is very ***skewed***



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., $r * f \approx k$ or $r * P_r \approx c$, where P_r is probability of word occurrence for the rth ranked word and $c \approx 0.1$ for English

Zipf’s Law

- Zipf’s Law relates a term’s frequency to its rank
  - frequency 1/rank
  - There is a constant *k* such that *freq* * *rank* = *k*
- The most frequent words in one corpus may be rare words in another corpus
  - Example: “computer” in CACM vs. National Geographic
- Each corpus has a different, fairly small “working vocabulary”

These properties hold in a wide range of languages

Zipf’s Law

- Useful as a rough description of the frequency distribution of words in human languages
- Behavior occurs in a surprising variety of situations
  - References to scientific papers
  - Web page in-degrees, out-degrees
  - Royalties to pop-music composers

Zipf’s Law (*Tom Sawyer*)

Word	Freq. (f)	Rank (r)	$f \cdot r$	Word	Freq. (f)	Rank (r)	$f \cdot r$
the	3332	1	3332	turned	51	200	10200
and	2972	2	5944	you'll	30	300	9000
a	1775	3	5235	name	21	400	8400
he	877	10	8770	comes	16	500	8000
but	410	20	8400	group	13	600	7800
be	294	30	8820	lead	11	700	7700
there	222	40	8880	friends	10	800	8000
one	172	50	8600	begin	9	900	8100
about	158	60	9480	family	8	1000	8000
more	138	70	9660	brushed	4	2000	8000
never	124	80	9920	sins	2	3000	6000
Oh	116	90	10440	Could	2	4000	8000
two	104	100	10400	Applausive	1	8000	8000

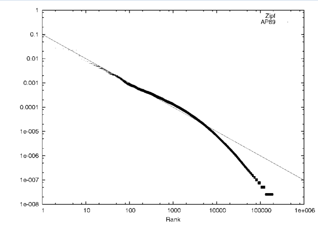
News Collection (AP89) Statistics

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

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 plot)



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 $1/n(n+1)$

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

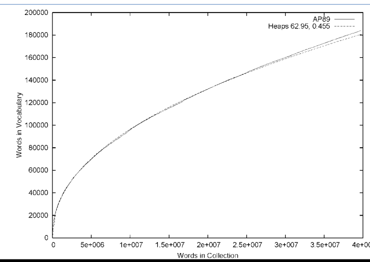
Vocabulary Growth

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

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

  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

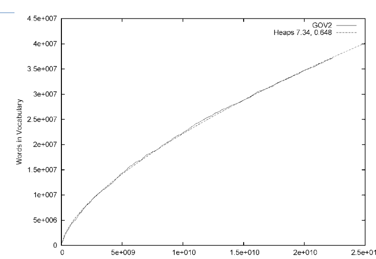
Heaps' Law: AP89 Example



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 corpora (i.e. < 1000 words) are much worse

GOV2 (Web) Example



Larger Corpora

- 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

Text transformation

- Word occurrence statistics
- Tokenizing
- Stopping and stemming

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 most search applications 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
  - 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
- Periods can occur in numbers, abbreviations, URLs, ends of sentences, and other situations
  - I.B.M., Ph.D., cs.umass.edu, F.E.A.R.
- Note: tokenizing steps for queries must be identical to steps for documents

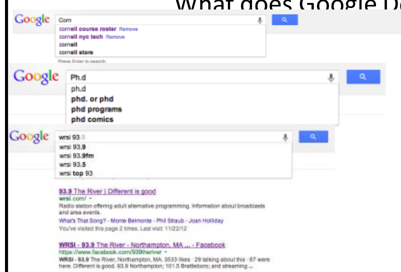
Tokenizing Process

- 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 one possibility is → 92 3 but search finds documents with 92 and 3 adjacent

Tokenizing Process

- Not that different than simple tokenizing process used in past
- Examples of rules sometimes used with TREC
  - Apostrophes in words ignored
    - o'connor → oconnor bob's → bobs
  - Periods in abbreviations ignored
    - I.B.M. → ibm Ph.D. → phd

What does Google Do?

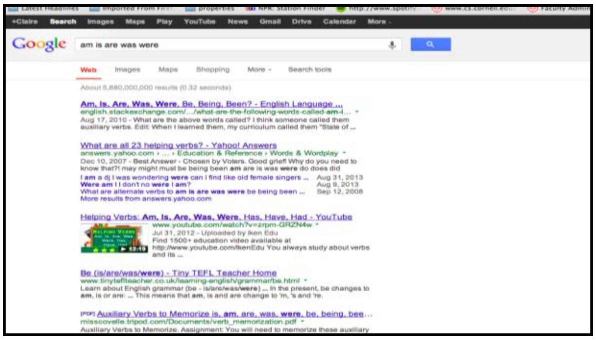


Stopping

- 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"

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
- 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 improvement in effectiveness
  - can be crucial for some languages
  - e.g., 5-10% improvement for English, up to 50% in Arabic

kitab	*a book*
kitabi	*my book*
alkitab	*the book*
kitabuki	*your book (f)*
kitabuka	*your book (m)*
kitabuhu	*his book*
kataba	*to write*
maktaba	*library, bookstore*
maktab	*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 positives*: supplies → supplie, ups → up
    - Some *false negatives*: mice → mice (should be mouse)

Lovins' stemmer

- For each word,
  - Find the longest suffix on the word
    - Check for exceptions for that suffix (go to next longest if one)
  - Remove the suffix
  - Check for a recode rule
    - Recode the remaining stem
      - believ -> belief
      - Revolv -> revolut

Beginning of suffix list

```
struct suftab {
 char *suf;                  /* actual suffix */
 short sufl;                 /* suffix length */
 short c_code;               /* condition code */
 }  suftab[] = {             /* actual suffix table as defined above */
     {"erentiations",12,36},     /* added by grb */
     {"alistically",11,1},
     {"antaneously",11,0},       /* added by grb */
     {"arizability",11,0},
     {"erentiation",11,36},      /* added by emv */
     {"izationally",11,1},
     {"antialness",10,0},
     {"arisations",10,0},
     {"arizations",10,0},
     {"entialness",10,0},
     {"entiations",10,0},        /* added by grb */
     {"ifications",10,25},       /* added by eaf */
```

Sample recode rules

```
case 0:   /* iev -> ief */
              *endword = 'f';
              break;
case 5:   /* olv -> olut */
              *endword++ = 'u';
              *endword++ = 't';
              *endword   = '\0';
              break;
```

Stemming (Manning et al examples)

- English: Such an analysis can reveal features … more biologically transparent
- Lovins: such an analys can reve featur … mor biolog transpar
- Porter: such an analysi can reveal feature … more biolog transpar
- Paice: such an analys can rev feat … mor biolog transp