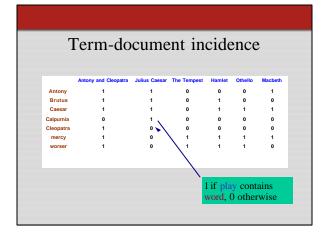
CS347

Lecture 1 April 4, 2001 ©Prabhakar Raghavan

Query

• Which plays of Shakespeare contain the words *Brutus AND Caesar* but *NOT Calpurnia*?



Incidence vectors So we have a 0/1 vector for each term. To answer query: take the vectors for *Brutus, Caesar* and *Calpurnia* (complemented) → bitwise *AND*. 110100 *AND* 110111 *AND* 101111 =

100100.

Answers to query

· Antony and Cleopatra, Act III, Scene ii

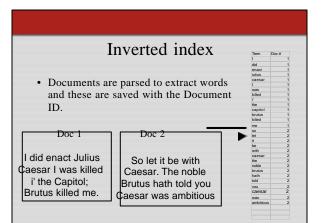
- Agrippa [Aside to DOMITIUS ENOBARBUS]: Why, Enobarbus, When Antony found Julius Caesar dead, He cried almost to roaring; and he we pt When at Philippi he found Brutusslain.
- Hamlet, Act III, Scene ii
- Lord Polonius: I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

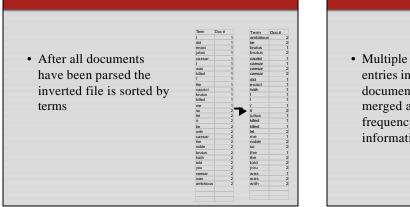
Bigger corpora

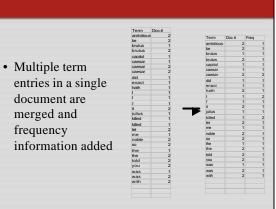
- Consider n = 1 documents, each with about 1K terms.
- Avg 6 bytes/term incl spaces/punctuation – 6GB of data.
- Say there are m = 500K <u>distinct</u> terms among these.

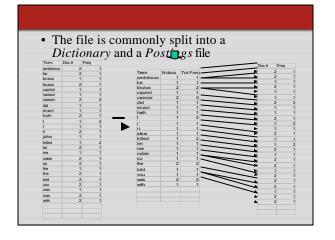
Can't build the matrix

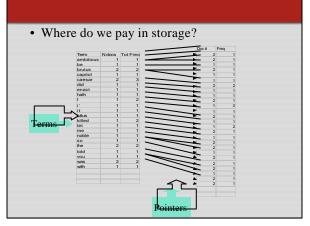
- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's. - matrix is extremely sparse.
- What's a better representation?











Two conflicting forces

- A term like *Calpurnia* occurs in maybe one doc out of a million - would like to store this pointer using log₂ 1M ~ 20 bits.
- A term like *the* occurs in virtually every doc, so 20 bits/pointer is too expensive.
 Prefer 0/1 vector in this case.

Postings file entry

- Store list of docs containing a term in increasing order of doc id.
 - Brutus: 33,47,154,159,202 ...
- <u>Consequence</u>: suffices to store gaps. - 33,14,107,5,43 ...
- <u>Hope</u>: most gaps encoded with far fewer than 20 bits.

Variable encoding

- For *Calpurnia*, use ~20 bits/gap entry.
- For *the*, use ~1 bit/gap entry.
- If the average gap for a term is G, want to use ~log₂G bits/gap entry.

γ codes for gap encoding

Length Offset

- Represent a gap G as the pair <*length*,*offset*>
- *length* is in unary and uses $\lfloor \log_2 G \rfloor + 1$ bits to specify the length of the binary encoding of
- offset = $G 2^{\lfloor \log_2 G \rfloor}$
- e.g., 9 represented as 1110001.
- Encoding G takes $2\lfloor \log_2 G \rfloor + 1$ bits.

What we've just done

- Encoded each gap as tightly as possible, to within a factor of 2.
- For better tuning (and a simple analysis) need some handle on the distribution of gap values.

Zipf's law

- The *k*th most frequent term has frequency proportional to 1/k.
- Use this for a crude analysis of the space used by our postings file pointers.

Rough analysis based on Zipf

- Most frequent term occurs in *n* docs - *n* gaps of 1 each.
- Second most frequent term in n/2 docs -n/2 gaps of 2 each ...
- *k*th most frequent term in n/k docs
 - -n/k gaps of k each use $2\log_2 k + l$ bits for each gap;
 - net of $\sim (2n/k) \cdot \log_2 k$ bits for *k*th most frequent term.

Sum over k from 1 to 500K

- Do this by breaking values of k into groups: group *i* consists of 2^{*i*-1} £ k < 2^{*i*}.
- Group *i* has 2^{i-1} components in the sum, Work out each contributing at most $(2ni)/2^{i-1}$.
- Summing over *i* from 1 to 19, we get a net estimate of 340Mbits ~45MB for our index.

Caveats

- This is not the entire space for our index:
 does not account for dictionary storage;
 as we get further, we'll store even more stuff in
- the index.Assumes Zipf's law applies to occurrence of terms in docs.
- All gaps for a term taken to be the same.
- Does not talk about query processing.

Issues with index we just built

- How do we process a query?
- What terms in a doc do we index?
 All words or only "important" ones?
- <u>Stopword</u> list: terms that are so common that they're ignored for indexing.
 - e.g., the, a, an, of, to ...
 - language-specific.

Repeat postings size calculation if 100 most frequent terms are not indexed

Issues in what to index

Cooper's concordance of Wordsworth was published in 1911. The applications of full-text retrieval are legion: they include résumé scanning, litigation support and searching published journals on-line.

- Cooper's vs. Cooper vs. Coopers.
- Full-text vs. full text vs. {full, text} vs. fulltext.
- Accents: *résumé* vs. *resume*.

Punctuation

- *Ne'er*: use language-specific, handcrafted "locale" to normalize.
- *State-of-the-art*: break up hyphenated sequence.
- U.S.A. vs. USA use locale.
- a.out

Numbers

- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- 100.2.86.144

Case folding

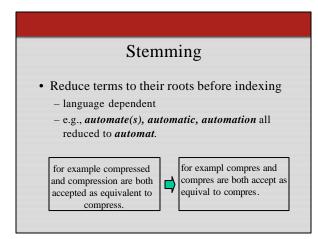
- Reduce all letters to lower case
 - proper nouns from language module
 - e.g., General Motors
 - Fed vs. fed
 - SAIL vs. sail

Thesauri and soundex

- Handle synonyms and homonyms
 - Hand-constructed equivalence classes
 - e.g., *car* = *automobile*
 - your→ you're
- Index such equivalences, or expand query? - More later ...

Spell correction

- Look for all words within (say) edit distance 3 (Insert/Delete/Replace) at query time
 - e.g., Alanis Morisette
- Spell correction is expensive and slows the query (upto a factor of 100)
 - Invoke only when index returns zero matches.
 - What if docs contain mis-spellings?



Porter's algorithm

- Commonest algorithm for stemming English
- Conventions + 5 phases of reductions
 - phases applied sequentially
 - each phase consists of a set of commands
 - sample convention: Of the rules in a compound command, select the one that applies to the longest suffix.

Typical rules in Porter

- $sses \rightarrow ss$
- $ies \rightarrow i$
- $ational \rightarrow ate$
- $tional \rightarrow tion$

So far: terms are the units of

- What about phrases?
- Proximity: Find Gates NEAR Microsoft.
- Need index to capture position information in docs.
- Zones in documents: Find documents with (*author* = Ullman) AND (text contains *automata*).

Evidence accumulation

- 1 vs. 0 occurrence of a search term - 2 vs. 1 occurrence
 - 3 vs. 2 occurrences, etc.
- Need term frequency information in docs

Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Need to measure proximity from query to each doc.
- Whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

Clustering and classification

- Given a set of docs, group them into clusters based on their contents.
- Given a set of topics, plus a new doc *D*, decide which topic(s) *D* belongs to.

The web and its challenges

- Unusual and diverse documents
- Unusual and diverse users, queries, information needs
- Beyond terms, exploit ideas from social networks
 - link analysis, clickstreams ...

Course administrivia

- Course URL: http://www.stanford.edu/class/cs347
- TA's: Taher Haveliwala, Brent Miller, Sriram Raghavan
- Grading:
 - 30% from midterm
 - 40% from final
 - 30% from group project.

Group project

- Groups of 4-5
- Strongly encouraged to build apps, not search engines
- Strongly encouraged to use one of the local corpora
- · Short and not onerous on programming
- Details to be updated on course page - Lead: Brent Miller; Discussion 4/11TBA

Class schedule

- Lectures MW 1250-205pm, Thornton 102
- April 11 guest lecture by Dr. Andrei Broder, Chief Scientist at Altavista (**) p
- Apr 30 mid-term in class
- May 23 onwards Distributed Databases, Sriram Raghavan lecturing

Resources for today's lecture

- Managing Gigabytes, Chapter 3.
- Modern Information Retrieval, Chapter 7.2
- Porter's stemmer: http://www.sims.berkeley.edu/~hearst/irbook/porter.html
- Shakespeare: http://www.theplays.org