CS276A

Text Information Retrieval, Mining, and Exploitation

Lecture 1

## Query

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- Could grep all of Shakespeare's plays for ***Brutus*** and ***Caesar*** then strip out lines containing ***Calpurnia***?
  - Slow (for large corpora)
  - *NOT* is non-trivial
  - Other operations (e.g., find the phrase ***Romans and countrymen***) not feasible



## Term-document incidence

| | Antony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth |
|---|---|---|---|---|---|---|
| Antony | 1 | 1 | 0 | 0 | 0 | 1 |
| Brutus | 1 | 1 | 0 | 1 | 0 | 0 |
| Caesar | 1 | 1 | 0 | 1 | 1 | 1 |
| Calpurnia | 0 | 1 | 0 | 0 | 0 | 0 |
| Cleopatra | 1 | 0 | 0 | 0 | 0 | 0 |
| mercy | 1 | 0 | 1 | 1 | 1 | 1 |
| worser | 1 | 0 | 1 | 1 | 1 | 0 |

1 if play contains word, 0 otherwise



## 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 wept
- When at Philippi he found ***Brutus*** slain.

- 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* = 1M documents, each with about 1K terms.
- Avg 6 bytes/term incl spaces/punctuation
  - 6GB of data.
- Say there are *m* = 500K *distinct* 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. Why?
  - matrix is extremely sparse.
- What's a better representation?



## Inverted index

- Documents are parsed to extract words and these are saved with the Document ID.

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

| Term | Doc # |
|---|---|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 2 |
| brutus | 2 |
| hath | 2 |
| told | 2 |
| you | 2 |
| caesar | 2 |
| was | 2 |
| ambitious | 2 |

- After all documents have been parsed the inverted file is sorted by terms

| Term | Doc # |
|---|---|
| I | 1 |
| did | 1 |
| enact | 1 |
| julius | 1 |
| caesar | 1 |
| I | 1 |
| was | 1 |
| killed | 1 |
| i' | 1 |
| the | 1 |
| capitol | 1 |
| brutus | 1 |
| killed | 1 |
| me | 1 |
| so | 2 |
| let | 2 |
| it | 2 |
| be | 2 |
| with | 2 |
| caesar | 2 |
| the | 2 |
| noble | 2 |
| brutus | 2 |
| hath | 2 |
| told | 2 |
| you | 2 |
| caesar | 2 |
| was | 2 |
| ambitious | 2 |

→

| Term | Doc # |
|---|---|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |



- Multiple term entries in a single document are merged and frequency information added

| Term | Doc # |
|---|---|
| ambitious | 2 |
| be | 2 |
| brutus | 1 |
| brutus | 2 |
| capitol | 1 |
| caesar | 1 |
| caesar | 2 |
| caesar | 2 |
| did | 1 |
| enact | 1 |
| hath | 1 |
| I | 1 |
| I | 1 |
| i' | 1 |
| it | 2 |
| julius | 1 |
| killed | 1 |
| killed | 1 |
| let | 2 |
| me | 1 |
| noble | 2 |
| so | 2 |
| the | 1 |
| the | 2 |
| told | 2 |
| you | 2 |
| was | 1 |
| was | 2 |
| with | 2 |

→

| Term | Doc # | Freq |
|---|---|---|
| ambitious | 2 | 1 |
| be | 2 | 1 |
| brutus | 1 | 1 |
| brutus | 2 | 1 |
| capitol | 1 | 1 |
| caesar | 1 | 1 |
| caesar | 2 | 2 |
| did | 1 | 1 |
| enact | 1 | 1 |
| hath | 2 | 1 |
| I | 1 | 2 |
| i' | 1 | 1 |
| it | 2 | 1 |
| julius | 1 | 1 |
| killed | 1 | 2 |
| let | 2 | 1 |
| me | 1 | 1 |
| noble | 2 | 1 |
| so | 2 | 1 |
| the | 1 | 1 |
| the | 2 | 1 |
| told | 2 | 1 |
| you | 2 | 1 |
| was | 1 | 1 |
| was | 2 | 1 |
| with | 2 | 1 |



- The file is commonly split into a *Dictionary* and a *Postings* file

| Term | Doc # | Freq |
|---|---|---|
| ambitious | 2 | 1 |
| be | 2 | 1 |
| brutus | 1 | 1 |
| brutus | 2 | 1 |
| capitol | 1 | 1 |
| caesar | 1 | 1 |
| caesar | 2 | 2 |
| did | 1 | 1 |
| enact | 1 | 1 |
| hath | 2 | 1 |
| I | 1 | 2 |
| i' | 1 | 1 |
| it | 2 | 1 |
| julius | 1 | 1 |
| killed | 1 | 2 |
| let | 2 | 1 |
| me | 1 | 1 |
| noble | 2 | 1 |
| so | 2 | 1 |
| the | 1 | 1 |
| the | 2 | 1 |
| told | 2 | 1 |
| you | 2 | 1 |
| was | 1 | 1 |
| was | 2 | 1 |
| with | 2 | 1 |

→

| Term | N docs | Tot Freq |
|---|---|---|
| ambitious | 1 | 1 |
| be | 1 | 1 |
| brutus | 2 | 2 |
| capitol | 1 | 1 |
| caesar | 2 | 3 |
| did | 1 | 1 |
| enact | 1 | 1 |
| hath | 1 | 1 |
| I | 1 | 2 |
| i' | 1 | 1 |
| it | 1 | 1 |
| julius | 1 | 1 |
| killed | 1 | 2 |
| let | 1 | 1 |
| me | 1 | 1 |
| noble | 1 | 1 |
| so | 1 | 1 |
| the | 2 | 2 |
| told | 1 | 1 |
| you | 1 | 1 |
| was | 2 | 2 |
| with | 1 | 1 |

| Doc # | Freq |
|---|---|
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 2 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 2 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |



- Where do we pay in storage?

Terms

| Term | N docs | Tot Freq |
|---|---|---|
| ambitious | 1 | 1 |
| be | 1 | 1 |
| brutus | 2 | 2 |
| capitol | 1 | 1 |
| caesar | 2 | 3 |
| did | 1 | 1 |
| enact | 1 | 1 |
| hath | 1 | 1 |
| I | 1 | 2 |
| i' | 1 | 1 |
| it | 1 | 1 |
| julius | 1 | 1 |
| killed | 1 | 2 |
| let | 1 | 1 |
| me | 1 | 1 |
| noble | 1 | 1 |
| so | 1 | 1 |
| the | 2 | 2 |
| told | 1 | 1 |
| you | 1 | 1 |
| was | 2 | 2 |
| with | 1 | 1 |

| Doc # | Freq |
|---|---|
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 1 |
| 2 | 2 |
| 1 | 1 |
| 1 | 1 |
| 2 | 1 |
| 1 | 2 |
| 1 | 1 |
| 2 | 1 |
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |
| 2 | 1 |
| 1 | 1 |
| 2 | 1 |
| 2 | 1 |

Pointers

## Two conflicting forces

- A term like ***Calpurnia*** occurs in maybe one doc out of a million - would like to store this pointer using $\log_2$ 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.

13

## Postings file entry

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

14

## 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 ~$\log_2 G$ bits/gap entry.

15

## $\gamma$ 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.

16

## 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.

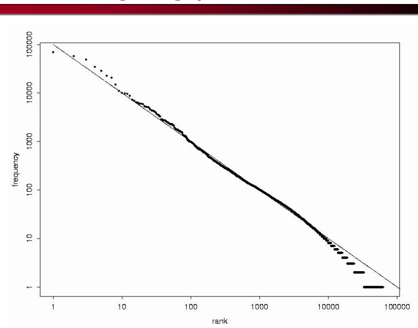


## Zipf's law

- 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.



## Zipf's law log-log plot






## 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 $2\log_2 k + 1$ bits for each gap;
  - net of ~$(2n/k).\log_2 k$ 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 $2^{i-1} \leq k < 2^i$.
- Group $i$ has $2^{i-1}$ components in the sum, 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.

Work out calculation.



## 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?
- Stopword list: terms that are so common that they're ignored for indexing.
  - *e.g.*, ***the, a, an, of, to*** …
  - language-specific.

*Exercise*: 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***

25

## Numbers

- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- 100.2.86.144
  - Generally, don't index as text
  - Creation dates for docs

26

## Case folding

- Reduce all letters to lower case
  - exception: upper case in mid-sentence
    - *e.g.,* ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***

27

## 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 ...

28

## 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?

29

## Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's*, *cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*

30

## Stemming

- Reduce terms to their “roots” before indexing
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compres and compres are both accept as equival to compres.

31

## 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.*

32

## Typical rules in Porter

- *sses* → *ss*
- *ies* → *i*
- *ational* → *ate*
- *tional* → *tion*

33

## Other stemmers

- Other stemmers exist, e.g., Lovins stemmer http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
- Single-pass, longest suffix removal (about 250 rules)
- Motivated by Linguistics as well as IR
- Full morphological analysis - modest benefits for retrieval

34

## Beyond term search

- 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***).

35

## 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

36

## 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.

37

## Structured vs unstructured data

- Structured data tends to refer to information in "tables"

| Employee | Manager | Salary |
|---|---|---|
| Smith | Jones | 50000 |
| Chang | Smith | 60000 |
| Ivy | Smith | 50000 |

Typically allows numerical range and exact match (for text) queries, e.g.,
*Salary < 60000 AND Manager = Smith.*

38

## Unstructured data

- Typically refers to free text
- Allows
  - Keyword queries including operators
  - More sophisticated "concept" queries e.g.,
    - find all web pages dealing with *drug abuse*
- Classic model for searching text documents

39

## Semi-structured data

- But in fact almost no data is "unstructured"
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
- Facilitates "semi-structured" search such as
  - *Title* contains data AND *Bullets* contain search

40

## More sophisticated semi-structured search

- *Title* is about Object Oriented Programming AND *Author* something like stro*rup
- where * is the wild-card operator
- Issues:
  - how do you process "about"
  - how do you rank results
- Will consider when studying XML search

41

## 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.
- Subject of CS276B next quarter.

42

## 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 ...

43

## 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

44

## Course administrivia

- Course URL: http://www.stanford.edu/class/cs276a/
- Grading:
  - 20% from midterm
  - 40% from final
  - 40% from project.

45

## Course staff

- **Professor:** Christopher Manning
  Office: Gates 418
  manning@cs.stanford.edu
- **Professor:** Prabhakar Raghavan
  pragh@db.stanford.edu
- **Professor:** Hinrich Schütze
  schuetze@csli.stanford.edu
- Office Hours: F 10-12
- **TA:** Taher Haveliwala
  Office: Gates B24A
  Office Hours: MW 1:30-3:00
  taherh@cs.stanford.edu

46

## Course project

- 40% of grade
- Groups of 2
- Don't build a search engine
  - Lucene engine available
- Watch for more details in Oct 3 lecture

47