Query

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

Term-document incidence

<table>
<thead>
<tr>
<th>Term</th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>venery</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Brutus AND Caesar but NOT Calpurnia

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) \( \land \) bitwise AND.
- \( 10100 \land 110111 \land 101111 = 100100 \).

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 \( n = 1 \text{M} \) documents, each with about 1K terms.
- Avg 6 bytes/term incl spaces/punctuation
  - 6GB of data in the documents.
- Say there are \( m = 500 \text{K} \) 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.
  - matrix is extremely sparse.
- What’s a better representation?
  - We only record the 1 positions.

Inverted index

- For each term $T$, we must store a list of all documents that contain $T$.
- Do we use an array or a list for this?

Brutus

Calpurnia

Caesar

What happens if the word Caesar is added to document 147?

Inverted index construction

Documents to be indexed.

Tokenizer

Linguistic modules

Indexer

Inverted index.

Indexer steps

- Sequence of (Modified token, Document ID) pairs.

Doc 1

Doc 2

I did enact Julius Caesar; I was killed in the Capitol. Brutus hath told you Caesar was ambitious.

So let it be with Caesar. The noble Brutus killed me.
Multiple term entries in a single document are merged.
Frequency information is added.

Why frequency? Will discuss later.

Where do we pay in storage?

Pointers
Terms

The index we just built

How do we process a query?
Later - what kinds of queries can we process?

Today’s focus

Query processing

Consider processing the query:
Brutus AND Caesar
- Locate Brutus in the Dictionary;
- Retrieve its postings.
- Locate Caesar in the Dictionary;
- Retrieve its postings.
- “Merge” the two postings:

Walk through the two postings simultaneously, in time linear in the total number of postings entries

If the list lengths are x and y, the merge takes $O(x+y)$ operations.
**Crucial**: postings sorted by docID.
Boolean queries: Exact match

- Boolean Queries are queries using AND, OR and NOT together with query terms
- Views each document as a set of words
- Is precise: document matches condition or not.
- Primary commercial retrieval tool for 3 decades.
- Professional searchers (e.g., lawyers) still like Boolean queries:
  - You know exactly what you’re getting.

Example: WestLaw  
http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- About 7 terabytes of data; 700,000 users
- Majority of users still use boolean queries
- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
- Long, precise queries; proximity operators; incrementally developed; not like web search

More general merges

- Exercise: Adapt the merge for the queries:
  - Brutus AND NOT Caesar
  - Brutus OR NOT Caesar

Can we still run through the merge in time O(x+y)?

Merging

What about an arbitrary Boolean formula?

- (Brutus OR Caesar) AND NOT (Antony OR Cleopatra)
- Can we always merge in “linear” time?
  - Linear in what?
  - Can we do better?

Query optimization

- What is the best order for query processing?
- Consider a query that is an AND of t terms.
- For each of the t terms, get its postings, then AND together.

| Brutus | 2 4 8 16 32 64 128 |
| Calpurnia | 1 2 3 5 8 16 24 32 |
| Caesar | 13 16 |

Query: Brutus AND Calpurnia AND Caesar

Query optimization example

- Process in order of increasing freq:
  - start with smallest set, then keep cutting further.

This is why we kept freq in dictionary

| Brutus | 2 4 8 16 32 64 128 |
| Calpurnia | 1 2 3 5 8 16 24 32 |
| Caesar | 13 16 |

Execute the query as (Caesar AND Brutus) AND Calpurnia
More general optimization

- e.g., \((\text{madding OR crowd}) \text{ AND (ignoble OR strife)})\]
- Get freq’s for all terms.
- Estimate the size of each OR by the sum of its freq’s (conservative).
- Process in increasing order of OR sizes.

Exercise

- Recommend a query processing order for

<table>
<thead>
<tr>
<th>Term</th>
<th>Freq</th>
</tr>
</thead>
<tbody>
<tr>
<td>eyes</td>
<td>213312</td>
</tr>
<tr>
<td>kaleidoscope</td>
<td>87009</td>
</tr>
<tr>
<td>marmalade</td>
<td>107913</td>
</tr>
<tr>
<td>skies</td>
<td>271658</td>
</tr>
<tr>
<td>tangerine</td>
<td>46653</td>
</tr>
<tr>
<td>trees</td>
<td>316812</td>
</tr>
</tbody>
</table>

Query processing exercises

- If the query is \(\text{friends AND romans AND (NOT countrymen)}\), how could we use the freq of \(\text{countrymen}\)?
- Exercise: Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?
- Hint: Begin with the case of a Boolean formula query: the each query term appears only once in the query.

Beyond term search

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

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.
Structured vs unstructured data

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

<table>
<thead>
<tr>
<th>Employee</th>
<th>Manager</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>Jones</td>
<td>50000</td>
</tr>
<tr>
<td>Chang</td>
<td>Smith</td>
<td>60000</td>
</tr>
<tr>
<td>Ivy</td>
<td>Smith</td>
<td>50000</td>
</tr>
</tbody>
</table>

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

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

Structured data has been the big commercial success [think, Oracle...] but unstructured data is now becoming dominant in a large and increasing range of activities [think, email, the web].

Semi-structured data

- 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
  - to say nothing of linguistic structure

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?
- The focus of XML search.

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 ...
Exercise

- Try the search feature at http://www.rhymezone.com/shakespeare/
- Write down five search features you think it could do better

Course administrivia

- Course URL: cs276a.stanford.edu [a.k.a., http://www.stanford.edu/class/cs276a/]
- Work/Grading:
  - Problem sets (2) 20%
  - Practical exercises (2) 20%
  - Midterm 20%
  - Final 40%
- Textbook:
  - No required text
  - Managing Gigabytes best early on
  - Will distribute brief readings

Looking ahead to CS276B (winter)

Course organization: two quarter sequence

- 276A Text Retrieval and Mining
  - We cover all the basic search and machine learning techniques for text
  - Small practical exercises; no big project
- 276B Web Search and Mining
  - Web search challenges
  - Link analysis, crawling, and other web-specifics
  - (Textual) XML
  - Project

Course staff

- Professor: Christopher Manning
  - Office: Gates 158 [new office!]
  - manning@cs.stanford.edu
- Professor: Prabhakar Raghavan
  - pragh@db.stanford.edu
- TA: Louis Eisenberg
  - tarheel@stanford.edu
- In general don’t use the above addresses, but:
  - su.class.cs276a (first option)
  - cs276a-aut0405-staff@lists.stanford.edu

Resources for today’s lecture

- Managing Gigabytes, Chapter 3.2
- Modern Information Retrieval, Chapter 8.2
- Shakespeare:
  - http://www.rhymezone.com/shakespeare/
  - Try the neat browse by keyword sequence feature!

Any questions?