Introduction to Information Retrieval

CS159
Spring 2011
David Kauchak

adapted from:
http://www.stanford.edu/class/cs276/handouts/lecture1-intro.ppt

Paper presentation guidelines

- Introduction
  - what is the problem
  - why do we care about it? why is it important?
- Background information
  - information not necessarily in the paper, but helps to understand the concepts
  - maybe some prior work (though for the length of these, you often don’t need to present this)
- Algorithm/approach
  - clearly spell out the approach
  - often useful to give a small example and walk through it

Administrative

- Partner/extra person for final project?
- E-mail me by the end of the day today
- If you’re a group of 2 and would like a 3rd person, e-mail me as well
- Read the articles

Experiments

- setup:
  - what is the specific problem?
  - what data are they using?
  - evaluation metrics?
- results
  - graphs/tables
  - analysis!
- Conclusions/future work
  - what have we shown/achieved?
  - where to now?
- Discussion
  - any issues with the paper?
  - any interesting future work?
  - interesting implications?
Paper presentation guidelines

- Misc
  - Presenting the material
    - be energetic/enthusiastic
    - make sure you know the material!
    - don’t read directly from your slides (or note cards if you bring them)
    - use some visual presentation software (e.g. powerpoint)
    - audience interaction is good (though not necessary for this type of presentation)
  - Avoid lots of text (i.e. this is a bad slide 😞)
    - powerpoint has a notes feature that you can use to remind yourself what you want to say, but not show to the audience (you can also print it out and use this instead)
  - use lots of images/figures/diagrams
  - show examples to illustrate algorithms/points
  - go beyond the paper – papers and presentations have difference goals

Information retrieval (IR)

- What comes to mind when I say “information retrieval”?

- Where have you seen IR? What are some real-world examples/uses?
  - Search engines
  - File search (e.g. OS X Spotlight, Windows Instant Search, Google Desktop)
  - Databases?
  - Catalog search (e.g. library)
  - Intranet search (i.e. corporate networks)

Web search

<table>
<thead>
<tr>
<th>Domain</th>
<th>September 2010</th>
<th>January 2011</th>
<th>February 2011</th>
<th>Month-over-Month-Point Change (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Sites</td>
<td>66.1</td>
<td>65.6</td>
<td>65.4</td>
<td>0.2</td>
</tr>
<tr>
<td>Yahoo Sites</td>
<td>16.7</td>
<td>16.1</td>
<td>16.1</td>
<td>0.0</td>
</tr>
<tr>
<td>Microsoft Bing</td>
<td>13.2</td>
<td>13.1</td>
<td>13.6</td>
<td>0.5</td>
</tr>
<tr>
<td>Ask Answer</td>
<td>3.7</td>
<td>3.4</td>
<td>3.2</td>
<td>-0.2</td>
</tr>
<tr>
<td>AOL Network</td>
<td>2.3</td>
<td>1.7</td>
<td>1.7</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Introducing Information Retrieval

Web search

![Web search traffic chart]

July 2006

February 2011

Challenges

- Why is information retrieval hard?
  - Lots and lots of data
  - Efficiency
  - Storage
  - Discovery (web)
- Data is unstructured
- Querying/Understanding user intent
- SPAM
- Data quality

Information Retrieval

- Information Retrieval is finding material in documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

Information Retrieval

- Information Retrieval is finding material in documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

?
Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content
  - Find all documents about computer science
  - Find all course web pages at Pomona
  - What is the cheapest flight from LA to NY?
  - Who is was the 15th president?

Information Retrieval

- Information Retrieval is finding material in text documents of an unstructured nature that satisfy an information need from within large collections of digitally stored content

What is the difference between an information need and a query?

IR vs. databases

- 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., Salary < 60000 AND Manager = Smith.
Introduction to Information Retrieval

Unstructured (text) vs. structured (database) data in 1996

The web

Web is just the start...

27 million tweets a day

247 billion e-mails a day
Challenges

- Why is information retrieval hard?
  - Lots and lots of data
    - efficiency
    - storage
    - discovery (web)
  - Data is unstructured
  - Understanding user intent
  - SPAM
  - Data quality

Efficiency

- 27 million tweets over 4 years = ~40 billion tweets
  - How much data is this?
    - ~4 TB of data uncompressed for the text itself
    - ~40 TB of data including additional meta-data
  - 40 billion web pages?
    - assume web pages are 100 times longer than tweets
      - 400 TB of data
      - 100 4 TB disks
    - assume web pages are 1000 times longer than tweets
      - 4 PB of data
      - 1000 4 TB disks
    - assume web pages are 10,000 times longer than tweets
      - 40 PB of data
      - 10,000 4 TB disks

Unstructured data in 1680

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

All of Shakespeare’s plays

How can we answer this query quickly?
Unstructured data in 1680

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

- **Key idea:** we can pre-compute some information about the plays/documents that will make queries much faster

- **What information do we need?**

- **Indexing:** for each word, keep track of which documents it occurs in

Inverted index

- For each term/word, store a list of all documents that contain it

- What data structures might we use for this?

  - **Brutus**
    - Array
    - Linked list
    - Hashtable

Inverted index construction

Documents to be indexed: Friends, Romans, countrymen

- **Text preprocessing:** friend, roman, countrymen

- **Indexer**

Inverted index

- **friend**
  - DocID: 2, 4

- **roman**
  - DocID: 1, 2

- **countryman**
  - DocID: 13, 16
**Boolean retrieval**

- Support queries that are boolean expressions:
  - A boolean query uses **AND**, **OR** and **NOT** to join query terms
    - Caesar **AND** Brutus **AND NOT** Calpurnia
    - Pomona **AND** College
    - (Mike **OR** Michael) **AND** Jordan **AND NOT** (Nike **OR** Gatorade)

- Given only these operations, what types of questions can't we answer?
  - Phrases, e.g. “Pomona College”
  - Proximity, “Michael” within 2 words of “Jordan”
  - Regular expression-like


- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of 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`
  - All words starting with “LIMIT”

**Boolean retrieval**

- Primary commercial retrieval tool for 3 decades
- Professional searchers (e.g., lawyers) still like boolean queries

- Why?
  - You know exactly what you’re getting, a query either matches or it doesn’t
  - Through trial and error, can frequently fine tune the query appropriately
  - Don’t have to worry about underlying heuristics (e.g. PageRank, term weightings, synonym, etc...)
Example: WestLaw  http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992)
- Tens of 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
  - /3 = within 3 words, /S = in same sentence

Query processing: AND

- What needs to happen to process: Brutus AND Caesar
- Locate Brutus and Caesar in the Dictionary;
  - Retrieve postings lists
- “Merge” the two postings:
  - Brutus AND Caesar 2\-8

The merge

- Walk through the two postings simultaneously
  - Brutus 2\-4\-8\-16\-32\-64\-128
  - Caesar 1\-2\-3\-5\-8\-13\-21\-34

Bron   AND  Caesar
The merge

- Walk through the two postings simultaneously

<table>
<thead>
<tr>
<th>Brutus</th>
<th>2 4 8 16 32 64 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caesar</td>
<td>1 2 3 5 8 13 21 34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brutus AND Caesar</th>
<th>2</th>
</tr>
</thead>
</table>

The merge

- Walk through the two postings simultaneously

<table>
<thead>
<tr>
<th>Brutus</th>
<th>2 4 8 16 32 64 128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caesar</td>
<td>1 2 3 5 8 13 21 34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Brutus AND Caesar</th>
<th>2</th>
</tr>
</thead>
</table>
The merge

- Walk through the two postings simultaneously

| Brutus | 2 - 4 - 8 - 16 - 32 - 64 - 128 |
| Caesar | 1 - 2 - 3 - 5 - 8 - 13 - 21 - 34 |

What is the running time?

\[ O(\text{length}_1 + \text{length}_2) \]

What assumption are we making about the postings lists?

For efficiency, when we construct the index, we ensure that the postings lists are sorted.

Boolean queries:
More general merges

- Which of the following queries can we still do in time \( O(\text{length}_1 + \text{length}_2) \)?

- Brutus AND NOT Caesar

- Brutus OR NOT Caesar
**From boolean to Google...**

- **What are we missing?**
  - Phrases
    - Pomona College
  - Proximity: Find *Gates NEAR Microsoft*.
  - Ranking search results
  - Incorporate link structure
  - Document importance

---

**Positional indexes**

- In the postings, store a list of the positions in the document where the term occurred

```
word1: 2 4 8 16
```

```
word2: (3, 16, 20) 4: (39) 8: (4, 18, 40) 16: (7)
```

```
docID: (position1, position2, ...)
```
Rank documents by text similarity

- Ranked information retrieval!
- Simple version: Vector space ranking (e.g. TF-IDF)
  - include occurrence frequency
  - weighting (e.g. IDF)
  - rank results by similarity between query and document
- Realistic version: many more things in the pot...
  - treat different occurrences differently (e.g. title, header, link text, ...)
  - many other weightings
  - document importance
  - spam
  - hand-crafted/policy rules

IR with TF-IDF

- How can we change our inverted index to make ranked queries (e.g. TF-IDF) fast?
- Store the TF initially in the index
- In addition, store the number of documents the term occurs in in the index
- IDF's
  - We can either compute these on the fly using the number of documents in each term
  - We can make another pass through the index and update the weights for each entry

From boolean to Google...

- Phrases
  - Pomona College
- Proximity: Find Gates NEAR Microsoft
- Ranking search results
  - include occurrence frequency
  - weighting
  - treat different occurrences differently (e.g. title, header, link text, ...)
- Incorporate link structure
- document importance

The Web as a Directed Graph

A hyperlink between pages denotes author perceived relevance AND importance

How can we use this information?
Query-independent ordering

- First generation: using link counts as simple measures of popularity
- Two basic suggestions:
  - **Undirected popularity:**
    - Each page gets a score = the number of in-links plus the number of out-links (3+2=5)
  - **Directed popularity:**
    - Score of a page = number of its in-links (3)

What is pagerank?

- The random surfer model
- Imagine a user surfing the web randomly using a web browser
- The pagerank score of a page is the probability that that user will visit a given page

Random surfer model

- We want to model the behavior of a “random” user interfacing the web through a browser
- Model is independent of content (i.e. just graph structure)
- What types of behavior should we model and how?
  - Where to start
  - Following links on a page
  - Typing in a url (bookmarks)
  - What happens if we get a page with no outlinks
  - Back button on browser
The questions...

- Given a graph and a teleporting probability, we have some probability of visiting every page
- What is that probability for each page in the graph?

Pagerank summary

- Preprocessing:
  - Given a graph of links, build matrix $P$
  - From it compute steady state of each state
  - An entry is a number between 0 and 1: the pagerank of a page
- Query processing:
  - Retrieve pages meeting query
  - Integrate pagerank score with other scoring (e.g. tf-idf)
  - Rank pages by this combined score

Pagerank problems?

- Can still fool pagerank
  - link farms
    - Create a bunch of pages that are tightly linked and on topic, then link a few pages to off-topic pages
  - link exchanges
    - I’ll pay you to link to me
    - I’ll link to you if you’ll link to me
  - buy old URLs
  - post on blogs, etc. with URLs
  - Create crappy content (but still may seem relevant)

IR Evaluation

- Like any research area, an important component is how to evaluate a system
- What are important features for an IR system?
- How might we automatically evaluate the performance of a system? Compare two systems?
- What data might be useful?
Measures for a search engine

- How fast does it index (how frequently can we update the index)
- How fast does it search
- How big is the index
- Expressiveness of query language
- UI
- Is it free?

- Quality of the search results

How do search engines make money?