Introduction to Information Retrieval

Adapted from:
http://www.stanford.edu/~cs276/handouts/lecture1-intro.ppt

Introductions

- Name/nickname
- Major and year
- One interesting thing about yourself
- Why are you taking this class?
- What topics/material would you like to see covered?
- Plans after graduation

Administrative

- go/cs458
- http://www.cs.middlebury.edu/~dkauchak/classes/cs458/
- Course overview
- Administrative

- TAing this semester?
- Homework 1 available later today (Due Tuesday)
- Programming assignment 1 available soon

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)
  - Search e-mail mailbox
**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.

**Diagram:**
- Document collection
- **R** system
- Query
- Answer list
- Info. need

**Dictionary says...**

- Oxford English Dictionary
  - **information**: informing, telling; thing told, knowledge, items of knowledge, news

- Random House Dictionary
  - **information**: knowledge communicated or received concerning a particular fact or circumstance; news

**Example Queries:**

- Find all documents about computer science
- Find all course web pages at Middlebury
- What is the cheapest flight from LA to NY?
- Who is the 15th president?
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.

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

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.

<table>
<thead>
<tr>
<th>Information need</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Find all documents about computer science</td>
<td>“computer science”</td>
</tr>
<tr>
<td>- Find all course web pages at Pomona</td>
<td>Pomona AND college AND url-contains class</td>
</tr>
<tr>
<td>- Who is was the 15th president?</td>
<td>WHO=president NUMBER=15</td>
</tr>
</tbody>
</table>

Challenges

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

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

200 million tweets/day over 4 years = ~300 billion tweets

How much data is this?
- ~40 TB of data uncompressed for the text itself
- ~400 TB of data including additional meta-data

300 billion web pages?
- assume web pages are 100 times longer than tweets
  - 4 PB of data
  - 1000 4 TB disks
- assume web pages are 1000 times long than tweets
  - 10,000 4 TB disks
- assume web pages are 10,000 times longer than tweets
  - 400 PB of data
  - 100,000 4TB disks

Unstructured data in 1680

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

You could grep all of Shakespeare’s plays for *Brutus* and *Caesar*, then strip out plays containing *Calpurnia*.

Any problems with this?
- Slow (for large corpora)
- Other operations (e.g., find the word *Romans* near *countrymen*) not feasible
- Ranked retrieval (best documents to return)
  - Later lectures

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
Introduction to Information Retrieval

Incidence vectors

For each term/word, we have a 0/1 vector

- Caeser = 110111
- Brutus = 110100
- Calpurnia = 010000

How can we get the answer from these vectors?

Term-document incidence matrix

For each term, we have a 0/1 vector

- Anto
- Brutus
- Caeser
- Calpurnia
- mercy
- wedge

1 if the play contains the word, 0 otherwise

Incidence vectors

For each term, we have a 0/1 vector

- Caeser = 110111
- Brutus = 110100
- Calpurnia = 010000

How can we get the answer from these vectors?

Bitwise AND the vectors together using the complemented vector for all NOT queries

Caeser AND Brutus AND COMPLEMENT(Calpurnia)

110111 & 110100 & ~010000 =
110111 & 110100 & 101111 =
100100
The answer

100100

Incidence vectors

For each term, we have a 0/1 vector
- Caesar = 110111
- Brutus = 110100
- Calpurnia = 010000

Bitwise AND the vectors together using the complemented vector for all NOT queries

Any problem with this approach?

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 collections

Consider $N = 1$ million documents, each with about 1000 words

Say there are $M = 500K$ distinct terms among these. How big is the incidence matrix?

The matrix is a 500K by 1 million matrix = half a trillion 0’s and 1’s
- Even for a moderate sized data set we can’t store the matrix in memory

Each vector has 1 million entries
- Bitwise operations become much more expensive
What does the matrix look like?

Consider $N = 1$ million documents, each with about 1000 words

Extremely sparse!

How many 1’s does the matrix contain?
- No more than one billion
- Each of the 1 million documents has at most 1000 1’s
- In practice, we’ll see that the number of unique words in a document is much less than this

Better representation?
- Only record the 1 positions

Inverted index

For each term, we store a list of all documents that contain it

What data structures might we use for this?

- Brutus
  - 2 4 8 16 32 64 128
    - array
  - 2 4 8 16 32 64 128
    - linked list
  - 4 128 32 64 28 16
    - hashtable

Pros
- Simple to implement
- No extra pointers required for data structure
- Contiguous memory

Cons
- How do we pick the size of the array?
- What if we want to add additional documents?

Inverted index representation

Pros
- Dynamic space allocation
- Insertion of new documents is straightforward

Cons
- Memory overhead of pointers
- Noncontiguous memory access
Inverted index representation

<table>
<thead>
<tr>
<th>Brutus</th>
<th>4</th>
<th>128</th>
<th>32</th>
<th>64</th>
<th>2</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>hashtable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Pros
- Search in constant time
- Contiguous memory

Cons
- How do we pick the size?
- What if we want to add additional documents?
- May have to rehash if we increase in size
- To get constant time operations, lots of unused slots/memory

Inverted index

The most common approach is to use a linked list representation

Dictionary
- Brutus
- Calpurnia
- Caesar

Postings lists
- Brutus
- Calpurnia
- Caesar

Boolean retrieval

In the boolean retrieval model we ask a query that is a boolean expression:
- A boolean query uses AND, OR and NOT to join query terms
  - Caesar AND Brutus AND NOT Calpurnia
  - Middlebury 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. “Middlebury College”
- Proximity, "Michael" within 2 words of "Jordan"
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
- All words starting with “LIMIT”

Example query:
- Requirements for disabled people to be able to access a workplace
- disabl/p acces/s work-site work-place (employment /3 place)

Long, precise queries; proximity operators; incrementally developed; not like web search

Professional searchers often like Boolean search:
- Precision, transparency and control

But that doesn’t mean it actually works better....
Query processing: AND

What needs to happen to process:
Brutus AND Caesar

Locate Brutus and Caesar in the Dictionary:
Retrieve postings lists

Brutus

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 8 & 16 & 32 & 64 & 128 \\
\end{array}
\]

Caesar

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 5 & 8 & 13 & 21 & 34 \\
\end{array}
\]

“Merge” the two postings:

Brutus AND Caesar

\[
\begin{array}{cccccccc}
2 & 8 \\
\end{array}
\]

The merge

Walk through the two postings simultaneously

Brutus

\[
\begin{array}{cccccccc}
2 & 4 & 8 & 16 & 32 & 64 & 128 \\
\end{array}
\]

Caesar

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 5 & 8 & 13 & 21 & 34 \\
\end{array}
\]

Brutus AND Caesar
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

Brutus AND Caesar 2
**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 assumption are we making about the postings lists?
- For efficiency, when we construct the index, we ensure that the postings lists are sorted.

What is the running time?
- $O(\text{length}_1 + \text{length}_2)$