Modern Information Retrieval

Introduction

Ruixuan Li
School of Computer Science & Technology
Huazhong University of Science & Technology
http://idc.hust.edu.cn/~rxli/
The Internet is enormous.
The 'Network Effect' kicks in, and the web goes critical'
Web Sites Increasing Sharply

- Number of web sites↑↑↑
  - 1993-1996, from 130 to 600,000 sites
  - 2008-2011, from 178,000,000 to 486,000,000
  - Netcraft
Web Sites Increasing Sharply

- Number of web sites↑↑↑
  - 1993-1996, from 130 to 600,000 sites
  - 2010-2013, from 200,000,000 to 780,000,000
  - Netcraft
CNNIC Report: The number of Internet users in China reached 253 million in June 2008, which has been ranked first in the world. The number of netizens reached 475 million at the end of 2010.
Current Web

1 billion users
more than 1000 billion pages

Static
WWW
URI, HTML, HTTP
Information Overload

- 1~2 EB (1 EB ≈ 10^{18} B) each year, which means 250MB per person on the earth
- Tens of billions of static web pages, and thousands of billions of dynamic web pages
- However, human brain can only store 200MB information, and deal with 6GB data in his whole life -- Tom Landauer
Problems

- How to understand the meanings of the information
- How to understand the user’s requirements
- How to fast and accurately find the information exactly matching the user’s requirements
Outline

- Text Information Management
- Information Retrieval
- Related Areas and Tools
- Related Research
Coverage of Topics: IR vs. TIM

IR and TIM will be used interchangeably

Text Information Management (TIM)

Information Retrieval (IR)

Multimedia, etc
Text Management Applications

- Access
- Select information
- Mining
- Create Knowledge
- Organization
- Add Structure/Annotations
Elements of Text Information Management Technologies

- Retrieval Applications
  - Summarization
  - Filtering
  - Search
  - Categorization

- Information Access
  - Information Organization
    - Natural Language Content Analysis

- Mining Applications
  - Visualization
  - Mining
  - Extraction
  - Clustering

Focus of this course
Text Management Hierarchy

User

TM Applications

TM Algorithms

Text

- Human-computer interaction
- Software engineering
- Web
- Probabilistic inference
- Machine learning
- Natural language processing
- Storage
- Compression

Information Science

Computer Science
Examples of Text Management Apps

- **Search**
  - Web search engines (Google, Yahoo, Baidu, …)
  - Library systems
  - …

- **Recommendation**
  - News filter
  - Literature/movie recommender

- **Categorization**
  - Automatically sorting emails
  - …

- **Mining/Extraction**
  - Discovering major complaints from email in customer service
  - Business intelligence
  - Bioinformatics
  - …

- **Many others…**
Outline

- Text Information Management
- Information Retrieval
- Related Areas and Tools
- Related Research
A Glance of IR System

1. Result1
2. Result2
3. Result3
Architecture of IR System

Diagram:

- User Interface
- Text
- Text Collection
- Collection Manager
- Query
- Query R
- Ranked Docs
- User Feedback
- Query Operations
- Indexing
- Text R
- Retrieved Docs
- Searching
- Ranking

Logical View:

1. User Interface
2. Text Operations
3. Text Collection
4. Collection Manager
5. Query
6. Query R
7. Ranked Docs
8. User Feedback
9. Query Operations
10. Indexing
11. Text R
12. Retrieved Docs
13. Searching
14. Ranking
Information Retrieval

- Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).
Unstructured (text) vs. structured (database) data in 1996

- Data volume
- Market Cap

Chart showing comparison between unstructured and structured data.
Unstructured (text) vs. structured (database) data in 2009
Unstructured data in 1680

- 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?

- Why is that not the answer?
  - Slow (for large corpora)
  - NOT Calpurnia is non-trivial
  - Other operations (e.g., find the word Romans near countrymen) not feasible
  - Ranked retrieval (best documents to return)
## 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>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>worser</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

1 if play contains word, 0 otherwise

**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) \( \bowtie \) bitwise AND.
- \( 110100 \text{ AND } 110111 \text{ AND } 101111 = 100100. \)
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.
Basic assumptions of Information Retrieval

- **Collection**: Fixed set of documents
- **Goal**: Retrieve documents with information that is relevant to the user’s information need and helps the user complete a task
The classic search model

TASK
Info Need
Verbal form
Info about removing mice without killing them
How do I trap mice alive?
Mistranslation?
Misconception?
Misformulation?
Find this: mouse trap any language
Search

Query
SEARCH ENGINE
Results
Corpus
Query Refinement
How good are the retrieved docs?

- **Precision**: Fraction of retrieved docs that are relevant to user’s information need

- **Recall**: Fraction of relevant docs in collection that are retrieved

- More precise definitions and measurements to follow in later lectures
Bigger collections

- Consider $N = 1$ million documents, each with about 1000 words.
- Avg 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents.
- 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.
  - matrix is extremely sparse.
- What’s a better representation?
  - We only record the 1 positions.

Why?
Inverted index

- For each term \( t \), we must store a list of all documents that contain \( t \).
  - Identify each by a docID, a document serial number

- Can we use fixed-size arrays for this?

<table>
<thead>
<tr>
<th>Brutus</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>11</th>
<th>31</th>
<th>45</th>
<th>173</th>
<th>174</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caesar</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>16</td>
<td>57</td>
<td>132</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>2</td>
<td>31</td>
<td>54</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What happens if the word \textit{Caesar} is added to document 14?
Inverted index

- We need variable-size postings lists
  - On disk, a continuous run of postings is normal and best
  - In memory, can use linked lists or variable length arrays
    - Some tradeoffs in size/ease of insertion

Sorted by docID (more later on why).
Inverted index construction

Documents to be indexed

Token stream

More on these later.

Linguistic modules

Modified tokens

Inverted index

Friends, Romans, countrymen.

Tokenizer

Indexer

friend

roman

countryman

friend

roman

countryman
Indexer steps: Token sequence

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

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
Indexer steps: Sort

- Sort by terms
  - And then docID

Core indexing step
Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

Why frequency? Will discuss later.
Where do we pay in storage?

<table>
<thead>
<tr>
<th>term</th>
<th>doc freq</th>
<th>postings lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>ambitious</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>be</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>brutus</td>
<td>2</td>
<td>→ 1 → 2</td>
</tr>
<tr>
<td>capitol</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>caesar</td>
<td>2</td>
<td>→ 1 → 2</td>
</tr>
<tr>
<td>did</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>enact</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>hath</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>i</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>i'</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>it</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>julius</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>killed</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>let</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>me</td>
<td>1</td>
<td>→ 1</td>
</tr>
<tr>
<td>noble</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>so</td>
<td>1</td>
<td>→ 2 → 2</td>
</tr>
<tr>
<td>the</td>
<td>2</td>
<td>→ 1 → 2</td>
</tr>
<tr>
<td>told</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>you</td>
<td>1</td>
<td>→ 2</td>
</tr>
<tr>
<td>was</td>
<td>2</td>
<td>→ 1 → 2</td>
</tr>
<tr>
<td>with</td>
<td>1</td>
<td>→ 2</td>
</tr>
</tbody>
</table>

Lists of docIDs

Later in the course:
- How do we index efficiently?
- How much storage do we need?
The index we just built

- How do we process a query?
  - Later - what kinds of queries can we process?
Query processing: AND

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:

```
Brutus
2 4 8 16 32 64 128

Caesar
1 2 3 5 8 13 21 34
```
The merge

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

If list lengths are $x$ and $y$, merge takes $O(x+y)$ operations.

**Crucial**: postings sorted by docID.
Intersecting two postings lists (a "merge" algorithm)

\[
\text{INTERSECT}(p_1, p_2)
\]

1. \( \text{answer} \leftarrow \langle \rangle \)
2. \( \textbf{while} \ p_1 \neq \text{NIL} \ \text{and} \ p_2 \neq \text{NIL} \)
3. \( \textbf{do} \ \textbf{if} \ \text{docID}(p_1) = \text{docID}(p_2) \)
4. \( \quad \textbf{then} \ \text{ADD}(\text{answer}, \text{docID}(p_1)) \)
5. \( \quad p_1 \leftarrow \text{next}(p_1) \)
6. \( \quad p_2 \leftarrow \text{next}(p_2) \)
7. \( \textbf{else} \ \textbf{if} \ \text{docID}(p_1) < \text{docID}(p_2) \)
8. \( \quad \text{then} \ p_1 \leftarrow \text{next}(p_1) \)
9. \( \quad \text{else} \ p_2 \leftarrow \text{next}(p_2) \)
10. \( \textbf{return} \ \text{answer} \)
The **Boolean retrieval model** is being able to ask a query that is a Boolean expression:

- Boolean Queries use **AND**, **OR** and **NOT** to join query terms
  - Views each document as a set of words
  - Is precise: document matches condition or not.
- Perhaps the simplest model to build an IR system on

- **Primary commercial retrieval tool for 3 decades.**
- **Many search systems you still use are Boolean:**
  - Email, library catalog, Mac OS X Spotlight
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
Example: WestLaw
http://www.westlaw.com/

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

- Note that SPACE is disjunction, not conjunction!

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

- Many professional searchers still like Boolean search
  - You know exactly what you are getting

- But that doesn’t mean it actually works better….
Boolean queries: More general merges

- **Exercise**: Adapt the merge for the queries:
  
  \[ \text{Brutus AND NOT Caesar} \]
  
  \[ \text{Brutus OR NOT Caesar} \]
  
  Can we still run through the merge in time \( O(x+y) \)?
  
  What can we achieve?
Merging

What about an arbitrary Boolean formula?

\[(\text{Brutus} \ OR \ \text{Caesar}) \ AND \ NOT \ (\text{Antony} \ OR \ \text{Cleopatra})\]

- 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 $n$ terms.
- For each of the $n$ terms, get its postings, then AND them together.

**Brutus**

1 2 4 8 16 32 64 128

**Caesar**

1 2 3 5 8 16 21 34

**Calpurnia**

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 document freq. in dictionary

Brutus

Caesar

Calpurnia

Execute the query as \((\text{Calpurnia AND Brutus}) \text{ AND Caesar}\).
More general optimization

- e.g., \((madding \ OR \ crowd) \ AND \ (ignoble \ OR \ strife)\)

- Get doc. freq.'s for all terms.

- Estimate the size of each \(OR\) by the sum of its doc. freq.'s (conservative).

- Process in increasing order of \(OR\) sizes.
Exercise

- Recommend a query processing order for

(tangerine OR trees) AND
(marmalade OR skies) AND
(kaleidoscope OR eyes)

<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

- **Exercise**: If the query is `friends AND romans AND (NOT countrymen)`, how could we use the freq of `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 where each term appears only once in the query.
Exercise

- Try the search feature at http://www.rhymezone.com/shakespeare/
- Write down five search features you think it could do better
What’s ahead in IR? Beyond term search

- What about phrases?
  - *Stanford University*

- Proximity: Find *Gates NEAR Microsoft*.
  - Need index to capture position information in docs.

- Zones in documents: Find documents with
  \[ \text{author} = \textbf{Ullman} \] \text{AND} \text{(text contains automata)}.\]
Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
  - Usually more seems better
- Need term frequency information in docs
Ranking search results

- Boolean queries give inclusion or exclusion of docs.

- Often we want to rank/group results
  - Need to measure proximity from query to each doc.
  - Need to decide whether docs presented to user are singletons, or a group of docs covering various aspects of the query.
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.,

\[ Salary < 60000 \text{ AND Manager} = Smith. \]
Unstructured data

- Typically refers to free-form 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
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 (*IIR* chapter 10)
Clustering, classification and ranking

- **Clustering**: Given a set of docs, group them into clusters based on their contents.

- **Classification**: Given a set of topics, plus a new doc $D$, decide which topic(s) $D$ belongs to.

- **Ranking**: Can we learn how to best order a set of documents, e.g., a set of search results
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 ...
- How do search engines work? And how can we make them better?
More sophisticated information retrieval

- Cross-language information retrieval
- Question answering
- Summarization
- Text mining
- …
Outline

- Text Information Management
- Information Retrieval
- Related Areas and Tools
- Related Research
Publications/ Societies

Learning/Mining
- ICML
- ICDM
- ICML, NIPS, UAI

Applications
- ISMB
- WWW
- RECOMB, PSB

Info. Science
- ACM SIGKDD
- IJCAI
- AAAI
- NLP
- HLT
- ACL
- COLING, EMNLP, ANLP

Info Retrieval
- ACM SIGIR
- ACM CIKM
- TREC

Databases
- ACM SIGMOD
- VLDB, PODS, ICDE

Software/systems
- SOSP
- OSDI

Statistics

(Incomplete)
Important Conferences

- SIGIR, WWW, SIGKDD, ACL
- CIKM, WSDM, ICDM, ICML, PKDD
- ECIR, TREC, PAKDD
- AIRS, WISE, WAIM, APWeb
Important Journals

- ACM Transactions on Information Systems (TOIS)
- ACM Transactions on Asian Language Information Processing (TALIP)
- IEEE Transactions on Knowledge and Data Engineering (TKDE)
- Information Processing & Management (IP&M)
- Information Retrieval
- World Wide Web Journal
Important Tools

- **Lucene**: full-featured text search engine library (Java) [http://lucene.apache.org/core/](http://lucene.apache.org/core/)
  - **Solr**: search platform (Java) [http://lucene.apache.org/solr/](http://lucene.apache.org/solr/)
  - **Nutch**: web-search software (Java) [http://nutch.apache.org/](http://nutch.apache.org/)
- **Larbin**: multi-purpose web crawler (C++) [http://larbin.sourceforge.net/index-eng.html](http://larbin.sourceforge.net/index-eng.html)
- **Weka**: data mining software (Java) [http://www.cs.waikato.ac.nz/ml/weka/](http://www.cs.waikato.ac.nz/ml/weka/)
- **Lemur**: information retrieval and text mining software (C++) [http://www.lemurproject.org/](http://www.lemurproject.org/)
- **Firtex**: full-text indexing and retrieval platform (C++) [http://www.firtex.org/](http://www.firtex.org/)
Outline

- Text Information Management
- Information Retrieval
- Related Areas and Tools
- Related Research
Research institutes and representatives

- **Salton** (1927-1995), Cornell Univ.
  - The founder of modern IR
  - SMART
  - The 1st Salton Award
  - ACM Fellow

- **Karen S. Jones** (1935-2007), Cambridge Univ.
  - Probabilistic retrieval model
  - Pioneer: NLP, IR
  - ACL Lifetime Achievement Award
  - Salton Award
Research institutes and representatives

- **W. Bruce Croft**, ACM Fellow, Umass CIIR
  - Statistical language modeling IR
  - Lemur (with CMU)
  - Salton Award

- **Rijsbergen**, ACM Fellow, Glasgow U., UK
  - Logical reasoning for Information Retrieval
  - IR using Quantum Physics
  - Salton Award

- **Robertson**, Microsoft Research Cambridge, London Metropolitan University, UK
  - Probabilistic retrieval model
  - OKAPI
  - Salton Award
Research institutes and representatives

- **Yiming Yang**, Language Technologies Institute, CMU
  - Text classification

- **Chengxiang Zhai**, UIUC
  - IR model

- **Jian-Yun Nie**, Université de Montréal
  - Cross language IR
  - IR model
Research institutes and representatives

- Weiying Ma, Microsoft Research Asia
- IBM Research Center
- Google
- Yahoo!
- Baidu
Research institutes and representatives (national)

- Peking University
- Harbin Institute of Technology
- Shanghai Jiao Tong University
- Tsinghua University
- Fudan University
- HUST
- Institute of Computing Technology, CAS
- Institute of Software, CAS
- Institute of Automation, CAS
- ......
Questions?