CIS 833 – Information Retrieval and Text Mining

Lecture 5

Retrieval Models

September 8, 2015

Credits for slides: Hofmann, Mihalcea, Mobasher, Mooney, Schutze.

MapReduce Exercise

- You have millions of records corresponding to user comments to various blog articles. Each record contains:

  (blog-article-ID, comment-article-ID, user-ID, comment-date, comment-message)

You are asked to use MapReduce to create a list of users with a per-user count of the number of unique blog articles that the user commented on. The input to the mapper will be files containing tuples having the format described above. The output of the reducer will consist of pairs of the form (*user, number-of-unique-blog-articles-the-user-commented-on*).

```
class MAPPER

method MAP(record_id, record_content)
    blog-article-ID<-get_blog-article-ID_from_record_content
    user-ID<-get_user-ID_from_record_content
    EMIT (user-ID,blog-article-ID)

class REDUCER

method REDUCE(user-ID, list[blog-article-IDs])
    sort(list[blog-article-IDs])
    sorted_list_unique<-remove_duplicated_from_sorted_list
    count<-count_number_of_articles(sorted_list_unique)
    EMIT(user-ID, count)
```

MapReduce Exercise

- Next, you are asked to use the tabulation of (*user, number-of-unique-blog-articles-the-user-commented-on*) from the previous job to find the users with the highest number of blog articles that they commented on. In your solution, you should use the fact that MapReduce sorts by intermediate keys into the reducer groups.

```
//we are reading the file produced in i. – i.e., each line contains user, count
method MAP(line_id, line_content)
split_line_content_into_user_and_count
EMIT (count, user)

//MapReduce will sort by intermediate keys –> count, user pairs will be sorted by count if
//only one reducer is used
method REDUCE (count, list [user1,user2,…])      //intermediate key,value pairs
        // reduce is the identity function
        for each user in list [user1,user2,…]
                EMIT (user, count)
```

Last, you are asked to use MapReduce to compute the number of pairs of users who both commented on the same blog article (each pair of users should be counted only once). The input will be the original files of form

(blog-article-ID, comment-article-ID, user-ID, comment-date, comment-message)

and the output will be

(blog-article-ID, count-of-user-pairs).

```
method MAP(record_id, record_content)
    blog-article-ID<-get_blog-article-ID_from_record_content
    user-ID<-get_user-ID_from_record_content
    EMIT (blog-article-ID, user-ID)


method REDUCE(blog-article-ID, list[user-IDs])
    sort(list[user-IDs])
    sorted_list_unique<-remove_duplicated_from_sorted_list
    N<-length(sorted_list_unique)
    count<-N(N-1)/2
    EMIT(blog-article-ID, count)
```

Assignments

- HW1 will be posted tomorrow (due next Wed)
- A *warmup* WordCount MapReduce programming assignment will also be posted online tomorrow (due in two weeks)

Required Reading

- "Information Retrieval" textbook
  - Chapter 1: Boolean Retrieval
  - Chapter 2: Term Vocabulary and Posting Lists
  - Chapter 4: Index Construction

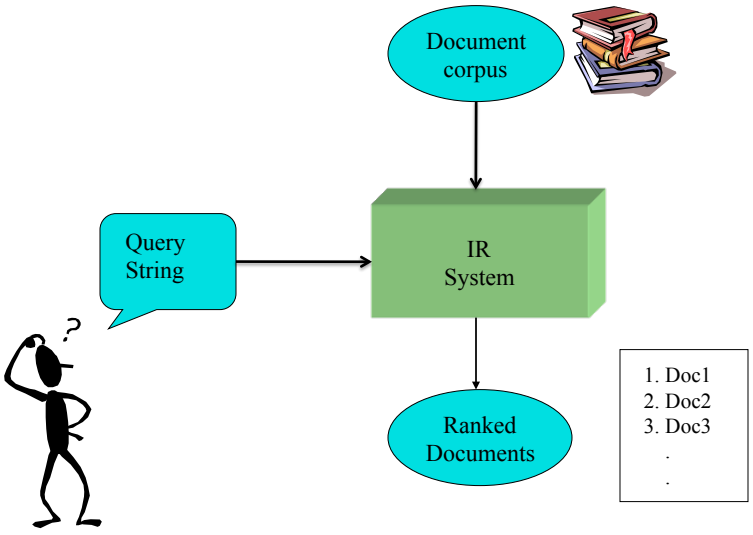
Information Retrieval

- The processing, indexing and retrieval of textual documents.
- Concerned firstly with retrieving *relevant* documents to a query.
- Concerned secondly with retrieving from *large* sets of documents *efficiently*.

Key Terms Used in IR

- Query: a representation of what the user is looking for - can be a list of words or a phrase.
- Document: an information entity that the user wants to retrieve
- Collection or corpus: a set of documents
- Index: a representation of information that makes querying easier
- Term: word or concept that appears in a document or a query

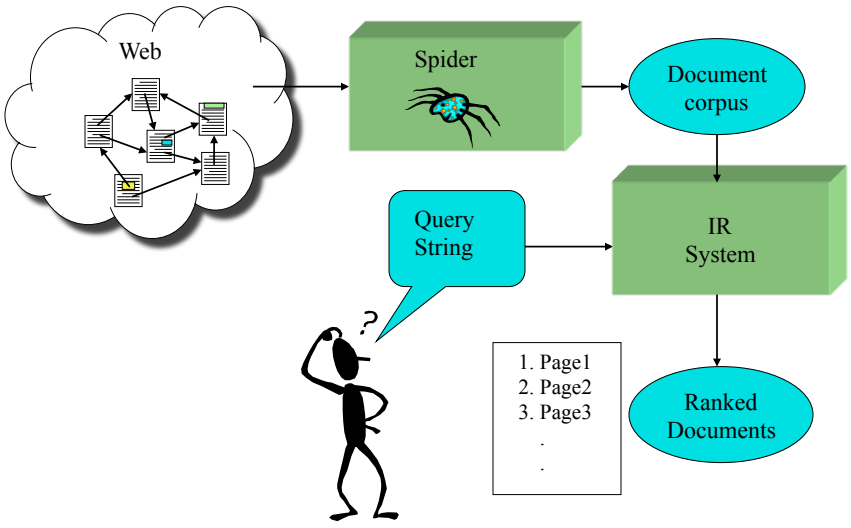
Typical IR System Architecture



Web Search

- Application of IR to HTML documents on the World Wide Web.
- Differences:
  - Must assemble document corpus by *spidering* the web.
  - Documents change uncontrollably.
  - Can exploit the structural layout information in HTML (or XML).
  - Can exploit the link structure of the web.

Web Search System



Typical IR Task

Given:

- A corpus of textual natural-language documents
- A user query in the form of a textual string

Find:

- A ranked set of documents that are relevant to the query

Relevance

- Relevance is a subjective judgment and may include:
  - Being on the proper subject.
  - Being timely (recent information).
  - Being authoritative (from a trusted source).
  - Satisfying the goals of the user and his/her intended use of the information (*information need*)
- Main relevance criterion: an IR system should fulfill user's information need

Basic IR Approach: Keyword Search

- Simplest notion of relevance is that the query string appears verbatim in the document.
- Slightly less strict notion is that the words in the query appear frequently in the document, in any order – bag of words representation
- How does this approach work?
  - Find words/concepts in documents
  - Compare them to words in a query
  - Very effective!

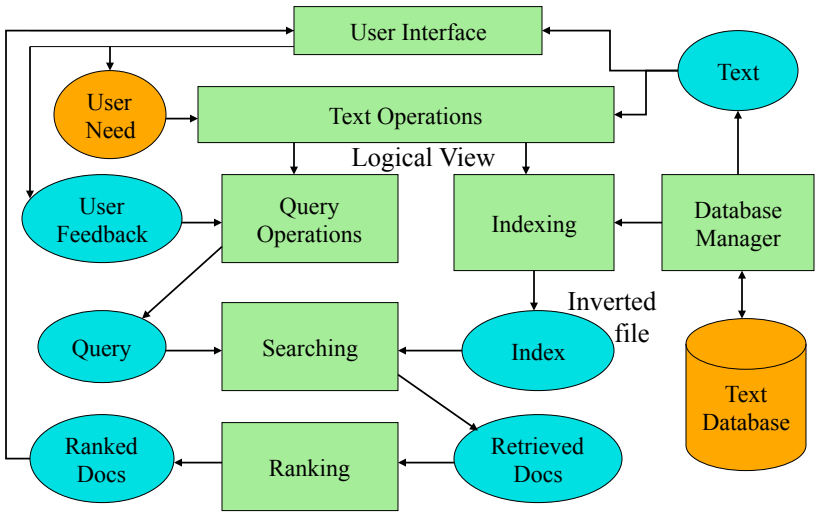
Problems with Keywords

- May not retrieve relevant documents that include synonymous terms.
  - "restaurant" vs. "café"
  - "PRC" vs. "China"
- May retrieve irrelevant documents that include ambiguous terms.
  - "bat" (baseball vs. mammal)
  - "Apple" (company vs. fruit)
  - "bit" (unit of data vs. act of eating)
- In this course:
  - We will cover the basics of keyword-based IR
  - Also, address more complex techniques for "intelligent" IR

Techniques for Intelligent IR

- Take into account the *meaning* of the words used
- Take into account the *order* of words in the query
- Adapt to the user based on automatic or semi-automatic *feedback*
- *Extend* search with related terms
- Perform automatic *spell checking* / *diacritics restoration*
- Take into account the *authority* of the source.

IR System Architecture



IR System Components

- Text Operations form index words (tokens)
  - Tokenization
  - Stop-word removal
  - Stemming
- Indexing constructs an inverted index of word to document pointers.
  - Mapping from keywords to document ids

Doc 1

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

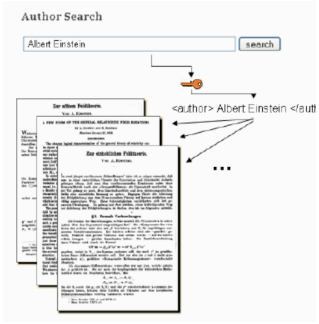
Doc 2

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

Document Indexing

- Index: associates a document with one or more keys (keywords)
- Present key → identify documents that match key
- Efficiency is crucial, fast access



Document Indexing: questions and challenges

- Two questions:
  - Which data structures and algorithms should be used for creating and maintaining indices?
  - Which attributes of documents should be utilized as keys?
- Conceptual challenges:
  - Usage: specify keys that are useful in searching and retrieving information
  - Creation: select attributes as keys that can be reliably determined from documents

Controlled Vocabulary vs Full-Text Indexing

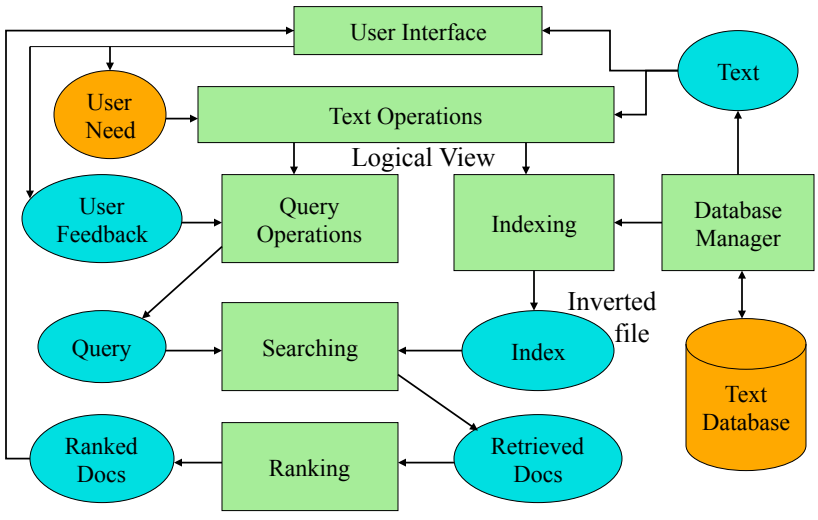
- Controlled vocabulary: list of terms whose meanings are specifically defined
  - terms need to be agreed upon
  - user needs to be aware of available vocabulary
  - approach pursued in the context of (digital) libraries – catalogues

- Full-text indexing: index terms are automatically selected from the document collection
  - documents need to be processed in order to extract index terms
  - selection heuristics for choosing appropriate terms
  - approach pursued by search engines

IR System Components

- Searching retrieves documents that contain a given query token from the inverted index.
- Ranking scores all retrieved documents according to a relevance metric.
- User Interface manages interaction with the user:
  - Query input and document output
  - Relevance feedback
  - Visualization of results
- Query Operations transform the query to improve retrieval:
  - Query expansion using a thesaurus
  - Query transformation using relevance feedback

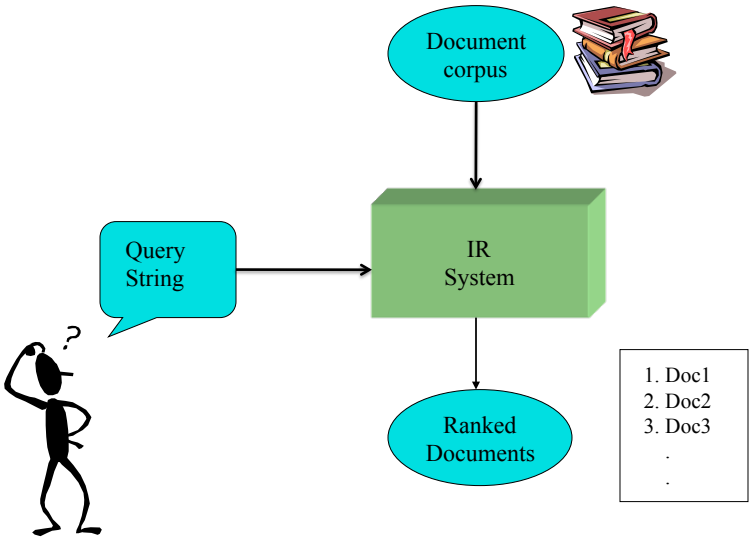
IR System Architecture



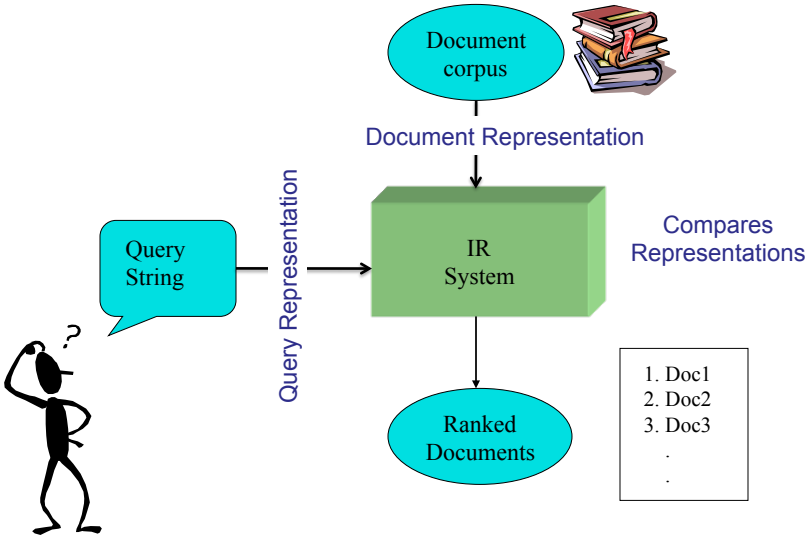
Retrieval Models

- A retrieval model specifies the details of:
  - Document representation
  - Query representation
  - How do we compare representations - retrieval function?
- Determines a notion of relevance.
- Notion of relevance can be binary or continuous (i.e., *ranked retrieval*).

Typical IR System Architecture



Typical IR System Architecture



Classes of Retrieval Models

- Boolean models (set theoretic)
  - Extended Boolean
- Vector space models (algebraic)
  - Generalized VS
  - Latent Semantic Indexing
- Probabilistic models
  - Inference Networks
  - Belief Networks

Exact match

Ranking - "Best" match