Chapter 1:
Intro to Information Retrieval

What is Information Retrieval?

Finding material (usually *documents*) of an *unstructured* nature (usually text) that satisfies an *information need* from within a large *collection*.

Define:

- Document - abstract grouping of content
- Unstructured - contrast with structured (database)
- Information need - a topic about which the user desires to know more
- Collection / CORPUS - a set of documents that may be unrelated

Contrast with database retrieval:

Suppose:

1. A database of CAMERAS for sale
2. Want to SEARCH for all cameras with $>$ 16 megapixels
3. How???

Brainstorm:

Examples of Information Retrieval systems

- Google
- Email
- Windows Search
- Grep - EXAMPLE ON PC

Grep:

- How to implement
- Problems
  - Large collections
  - More flexible searches
  - Ranked retrieval

Basic Indexing

Index - a data structure for associating occurrence in documents

Incidence matrix:

Collect the *terms* of all documents, build a *term-document matrix*.

Example 1. Shakespeare TD matrix from browser. http://nlp.stanford.edu/IR-book/html/htmledition/img38.png

- Answer the query `Antony`
- Answer the query `Brutus AND Caesar`
- Answer the query `Cleopatra OR Calpurnia`

- Answer the query `Brutus AND Caesar AND NOT Calpurnia`.

Boolean retrieval:

User specifies a **query** to communicate their **information need** in the form of **Boolean expression of terms**.

A boolean retrieval system returns all documents that satisfy the boolean query.

Assumed default operator is AND

Index size:

Example 2. A conservative estimate for the Library of Congress holdings is 20 million books. A conservative estimate of the union of all English terms in those books is 500,000. Provide a conservative estimate for the size (in bytes) of the associated Library of Congress term-document matrix.

Implementing the matrix:

1. Determine vocabulary – SORTED.
2. Create $T \times N$ matrix.
3. For each document:
   (a) Tokenize the text.
   (b) Do linguistic processing (normalization)
   (c) For each term, put a 1 in the matrix for the document and term

Effectiveness:

- A document is *relevant* if the user believes it satisfies the information need
- Precision - what fraction of returned results are relevant?
- Recall - what fraction of all relevant documents in the collection was returned?

Inverted Indexing

Problem: TD matrix is too damn big. **Observe**: the matrix is sparse (mostly 0's).

- Keep a **dictionary** of terms in the **vocabulary**.
- For each term, associate a **list** of which documents contain the term.
  - Each item in the list is called a **posting**; the list is a **postings list**

Building an inverted index:

1. Collect the documents to be indexed.
2. Tokenize the text.
3. Do linguistic processing (normalization)
4. For each token, add the document ID to the term's postings list

Data structures?

1. Want something to map from a string term to a list of integer IDs – HASH MAP

Size of inverted index:

Example 3. A conservative estimate for the Library of Congress holdings is 20 million books. A conservative estimate of the union of all English terms in those books is 500,000. What is the size of an inverted index over this collection?

Assume the frequency of the ith most frequent term in the vocabulary follows Zipf's law, $f_i = \frac{c}{i^s}$, with $c = s = 1$.

If each posting requires 4 bytes of space, and all we need is storage for the postings themselves (not the lists, or the vocabulary), then we have

$$4 \times \sum_{i=1}^{T} 20000000 \times f_i = 4 \times 20000000 \times \sum_{i=1}^{500000} \frac{1}{i} \approx 80000000 \times \ln(500000) \approx 1,049,789,070$$

Processing queries with inverted index

1. Boolean operators: AND, OR, NOT
2. Note: postings lists are in **increasing order by document ID**
3. Retrieve postings for X, for Y
4. Intersect postings
5. Repeat

Given the postings lists $l_1 = 1, 4, 7, 8, 10, 13, 20, 24, 25, 26, 29$ and $l_2 = 1, 5, 9, 11, 12, 13, 18, 24, 25, 28, 29, 40, 52$, determine resulting postings for l_1 AND l_2 , l_1 OR l_2, l_1 AND (NOT l_2).

Complexities of intersections:

- Upper bounds: AND $O(x+y)$, OR $O(x+y)$
- Lower bounds: AND $\Omega(\min(x,y))$, OR $\Omega(x+y)$

Order of processing conjunctive queries:

- Given postings of length x and y, what is the LARGEST list that results from x AND y, x OR y? What is the SMALLEST?

Is boolean retrieval good enough?

- Queries are FORMULAS that need training to use well
- Want to search compound words or phrases
- Only records PRESENCE (yes/no), but maybe we want documents with high frequency of terms
- Maybe we wnt to RANK documents on relevance