Information Retrieval

The typical information retrieval (IR) scenario:

- The user has an information need.

- The user specifies a query that describes the information need.

- The IR system retrieves a set of documents from the corpus that are believed to be relevant.

- The documents are often ranked according to their likelihood of being relevant.

Boolean Keyword Systems

- The user gives a set of keywords that are likely to appear in relevant documents.

  Example: to find documents about conspiracy theories regarding the assassination of JFK, you might use the keywords JFK, conspiracy, and assassination.

- Many systems support Boolean functions, such as AND and OR. So you could specify a query as:

\[
\text{AND(Kennedy, conspiracy, OR(assassination, murder))}
\]

  or

\[
\text{AND(OR(Kennedy,JFK),OR(conspiracy,plot), OR(assassination,assassinated,assassinate,murder,murdered,kill,killed))}
\]

Major Issues in Information Retrieval

Polysemy: many words have multiple meanings.

Synonymy: many different words can refer to the same concept.

Size: IR systems must be able to process huge volumes of text (often many gigabytes).

Broad Coverage: IR systems must be able to handle queries about any topic whatsoever!

Text Processing = Natural Language Processing?

- Many information retrieval systems claim that they do “natural language processing”. But this does not necessarily mean that they use syntax, semantics, or any methods aimed at real understanding.

  Almost always, this quite literally means that the system processes natural language documents.

- Sometimes an IR system will advertise that they accept natural language input. This just means that the user can type a query in English, rather than use a formal query language. In most cases, the system just extracts keywords from the query anyway!

- So beware!!! Many companies (and some researchers) rely on the fact that most people won’t understand the difference and will assume that the system is a lot more intelligent than it is!
Inverted Files

- Most IR systems use an inverted file to represent the documents in a corpus.

- An inverted file is essentially a table of words with a list of documents that contain each word.

  - assassination: (doc1, doc4, doc35, doc56, doc90, ...)
  - murder: (doc3, doc7, doc36, doc56, doc78, ...)
  - Kennedy: (doc24, doc27, doc29, doc44, doc90, ...)
  - conspiracy: (doc3, doc55, doc90, doc98, ...)

- Sometimes additional information is stored, such as the location of each word in the document. Proximity information allows a system to exactly or approximately match phrases.

Stopwords

- Most IR systems use a list of stopwords, which usually consists of closed class words that are not believed to contain much information content because they occur in so many documents.

- Stopwords are not added to the inverted file, which means they are essentially discarded.

- Removing stopwords greatly decreases the size of the inverted file.

  Typical Stopwords:
  - a, an, the
  - he, she, it, they, them, those, their, those, ...
  - of, to, from, by, with, for, at, ...
  - should, would, could, can, will, must, shall, ...

Disadvantages of Using Stopwords

- Sometimes common strings are used in uncommon ways. For example, a well-known IR system could not retrieve documents about a Vietnamese leader named “the”.

- Sometimes stopwords are an important component of a title or phrase. For example, you might want to retrieve documents that contain the famous Shakespeare quote: to be or not to be.

- Some stopwords, such as prepositions, provide crucial information about relationships.

- Disk space is much cheaper than it used to be, so saving space is not as important as it used to be!

Stemming

- Many IR systems use some form of stemming to match query terms with morphological variants in the documents.

  Example: if you use the word assassination, then it would be stemmed to produce the stem assassinat which would match:

  - assassination
  - assassinations
  - assassinate
  - assassinated
  - assassinating

- Some systems use a true morphological analyzer, but most stemmers just chop off common prefixes and suffixes.
Limitations of Stemming

<table>
<thead>
<tr>
<th>Errors by the Porter Stemmer</th>
<th>Omissions by the Porter Stemmer</th>
</tr>
</thead>
<tbody>
<tr>
<td>organization</td>
<td>organ</td>
</tr>
<tr>
<td>doing</td>
<td>doe</td>
</tr>
<tr>
<td>generalization</td>
<td>generic</td>
</tr>
<tr>
<td>numerical</td>
<td>numerous</td>
</tr>
<tr>
<td>policy</td>
<td>police</td>
</tr>
<tr>
<td>university</td>
<td>universe</td>
</tr>
<tr>
<td>easy</td>
<td>easily</td>
</tr>
<tr>
<td>addition</td>
<td>additive</td>
</tr>
<tr>
<td>negligible</td>
<td>negligent</td>
</tr>
<tr>
<td>execute</td>
<td>executive</td>
</tr>
<tr>
<td>define</td>
<td>definite</td>
</tr>
<tr>
<td>past</td>
<td>paste</td>
</tr>
<tr>
<td>ignore</td>
<td>ignorant</td>
</tr>
<tr>
<td>special</td>
<td>specialized</td>
</tr>
<tr>
<td>arm</td>
<td>army</td>
</tr>
<tr>
<td>head</td>
<td>heading</td>
</tr>
<tr>
<td>european</td>
<td>europe</td>
</tr>
<tr>
<td>analysis</td>
<td>analyzes</td>
</tr>
<tr>
<td>matrices</td>
<td>matrix</td>
</tr>
<tr>
<td>noise</td>
<td>noisy</td>
</tr>
<tr>
<td>sparse</td>
<td>sparsity</td>
</tr>
<tr>
<td>explain</td>
<td>explanation</td>
</tr>
<tr>
<td>resolve</td>
<td>resolution</td>
</tr>
<tr>
<td>triangle</td>
<td>triangular</td>
</tr>
<tr>
<td>urgency</td>
<td>urgent</td>
</tr>
<tr>
<td>cylinder</td>
<td>cylindrical</td>
</tr>
</tbody>
</table>

The Vector Space Model

- A document is commonly represented as an n-dimensional vector, where n is the size of the vocabulary.
- The original ordering of the terms in the document is ignored, which yields a **bag of words** representation.
- The documents in the text collection are viewed as a set of vectors in a vector space.
- A query is viewed as a vector in the same space. A vector-based similarity metric is used to find the document vector that is most similar to the query vector.

Cosine Similarity for IR

The most commonly used similarity metric is cosine similarity:

\[
\text{Cosine Similarity}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{|\vec{V}(d_1)||\vec{V}(d_2)|}
\]

The numerator is the dot (inner) product of two vectors. The denominator is the product of their Euclidean lengths.

The Euclidean length of \(d\) is defined as:

\[
\|d\| = \sqrt{\sum_{i=1}^{m} V_i^2(d)}
\]

For information retrieval, we treat a query as a (short!) document and measure the similarity between the query and a candidate document:

\[
\text{Similarity}(q, d) = \frac{\vec{V}(q) \cdot \vec{V}(d)}{|\vec{V}(q)||\vec{V}(d)|}
\]

Term Weighting

- The terms in a vector are weighted based on their perceived importance **with respect to the document**.
- A simple weighting scheme is called **Term Frequency**, \(TF(t,d)\), which counts the number of occurrences of term \(t\) in a document \(d\).
- But ... a problem with Term Frequency weights is that all words are assumed to be of equal importance.

For example, if a very general word (e.g., “thing”) and a very specific word (e.g., “cardiomyopathy”) each occur 5 times, they will both have the same weight of 5.
TF-IDF Weighting

TF-IDF is a widely used term weighting scheme that gives higher weights to terms that are less common, under the assumption that these words are particularly relevant when they do occur.

The TF-IDF formula uses Term Frequency (TF) and a measure called Inverse Document Frequency (IDF).

**Document Frequency, DF(t)**: the number of documents in the text collection that contain term \( t \).

**Inverse Document Frequency, IDF(t)**: \( \log\left(\frac{N}{DF(t)}\right) \) where \( N \) is the number of documents in the text collection.

TF-IDF \( (t,d) \) = TF \( (t,d) \) * IDF \( (t) \)

---

### TF-IDF Weighting Example

The table below shows results from a Reuters collection of 806,791 documents using \( \log_{10} \) IDF values.

<table>
<thead>
<tr>
<th>Word</th>
<th>DF(t)</th>
<th>IDF(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>best</td>
<td>25,235</td>
<td>1.50</td>
</tr>
<tr>
<td>car</td>
<td>18,165</td>
<td>1.65</td>
</tr>
<tr>
<td>auto</td>
<td>6,723</td>
<td>2.08</td>
</tr>
</tbody>
</table>

For the query “best auto insurance”, the term weights would be:

\( \text{best}=1.50, \text{auto}=2.08, \text{insurance}=0 \)

---

### Query Reformulation

- **Query Expansion** methods add new terms to a query to improve retrieval effectiveness.

- Existing thesauri or semantic knowledge bases (e.g., WordNet) can be used to add synonyms or semantically similar terms.

- Methods for automatic thesaurus generation can be used to expand the set of query terms.

- Spelling correction and similar cleaning techniques (e.g., phonetic correction) can be used to reformulate a query.

- All of these techniques often do improve performance, but pose some risk of shifting the query away from the user’s intention.

---

### Relevance Feedback

**Relevance Feedback (RF)** methods were designed to involve the user in a feedback process to improve retrieval performance. The typical steps are:

1. The IR system returns an initial set of documents.
2. The user labels the top \( k \) documents as being either relevant or nonrelevant.
3. The IR system uses the labels to improve its model.
4. The IR system produces a new set of documents, and the process repeats.
The Rocchio Algorithm

- The goal behind relevant feedback techniques is to reweight or add query terms to make the query more similar to the relevant documents and less similar to the nonrelevant documents.

- The Rocchio model is a widely used method for relevance feedback. $q_0$ is the original query vector, $D_r$ is the set of relevant documents, and $D_{nr}$ is the set of nonrelevant documents.

\[
\vec{q}_m = \alpha \vec{q}_0 + \beta \frac{1}{|D_r|} \sum_{d_j \in D_r} \vec{d}_j - \gamma \frac{1}{|D_{nr}|} \sum_{d_j \in D_{nr}} \vec{d}_j
\]

- $\alpha$, $\beta$, and $\gamma$ are weights that determine how much you want to trust the original query vs. the new labels.

- Relevance feedback is especially useful for increasing recall.

Pseudo Relevance Feedback

- Pseudo relevance feedback is a method for query modification that is fully automatic and does not rely on manual user judgements.

- First, an initial set of documents is retrieved. Then the top $k$ ranked documents are assumed to be relevant and the relevance feedback model is applied.

- This technique often improves performance, but generally isn’t as good as true relevance feedback. If the top $k$ documents are similar to each other in an undesirable way, the automatic process will reinforce that behavior and can drift from the original intent.

The State of the Art

The state of the art in IR is still relatively primitive from a natural language perspective!

Stopwords, stemming, and phrases are commonly used, and there are sophisticated algorithms for bag-of-words models. But not much true NLP is used. Why?

- Keywords and phrases are reasonably effective for many queries.
- Morphological conflation gets mixed results: it sometimes helps and sometimes hurts. Stemming performs about the same and is much easier.
- Syntactic processing has been tried and generally doesn’t help much.
- Semantic analysis holds potential, but it still poses challenges especially for broad-coverage IR tasks.