CSA4020

Multimedia Systems:
Adaptive Hypermedia Systems

Lecture 5:
Statistical Models
Of
Information Retrieval
Problems with Boolean Model

- Difficult to capture user information need and document content

  e.g., compare NL with SQL query for “retrieve documents that describe companies whose stock value has increased by 150% over the last 18 months”

- Difficult to rank output

- Difficult to control no. of docs retrieved

- Difficult to perform automatic relevance feedback

  user ranking vs relevance judgements vs “find more like this”
Blair & Maron’s 1985 study

• Tested Boolean Retrieval Model, STAIRS, to evaluate Precision and Recall

• Also wanted to test hypothesis that “terms” accurately predict information content of documents

• STAIRS indexed c. 40,000 legal documents

• Lawyer’s information request submitted to STAIRS by trained paralegals

• Results showed that although Precision was 80%, Recall was 20%

• Why might this be?
The Cosine Similarity Measure

\[ \text{sim}(Q, D_j) = \frac{\sum_{i=1}^{n} q_i w_{ij}}{\sqrt{\sum_{i=1}^{n} q_i^2 \sum_{i=1}^{n} w_{ij}^2}} \]

- Also allows us to visualise docs plotted into \( n \)-dimensional space
- Query can be plotted into same space
- Query’s nearest neighbours are most relevant documents...
- Can use same formula to find docs most similar to current doc...
- ... and to classify documents into categories (not too good...)
Implementation Issues

- Typically, vectors of docs in collection are stored in an inverted index file

More efficient: access, updates, etc

<table>
<thead>
<tr>
<th>Index terms</th>
<th>df</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>4</td>
</tr>
<tr>
<td>Computer</td>
<td>3</td>
</tr>
<tr>
<td>Database</td>
<td>1</td>
</tr>
<tr>
<td>Retrieval</td>
<td>3</td>
</tr>
</tbody>
</table>

Q=term₁, term₂, term₃, ...

D₁,4...
D₁,3...
D₁,6...
Document and Query Term Weights

\[ w_{i,j} = tf_{i,j} \cdot idf_j \]

- \( tf_{i,j} \) = frequency of term \( j \) in document \( i \)
- \( idf_j \) = inverse document frequency of term \( j \)

\[ idf_j = \log_2 \left( \frac{\text{number of documents}}{df_j} \right) \]

- \( df_j \) = document frequency of term \( j \)
  = number of documents containing term \( j \)

- term weight will be high in a doc if it appears frequently in doc, but infrequently in collection

- If weights not specified in query, choose from either \{0, 0.5\} or \{0,1\}

- NL query can be treated as doc
Normalising term weights

- We don’t want shorter documents to be considered more relevant than longer ones

- Assume the following:

  Longer docs contain more terms and/or more occurrences of the same terms than shorter docs, so:

  $freq_{i,j}$ = raw term freq of $T_{j}$ in $D_{i}$

  $f_{i,j}$ = normalised frequency of $T_{j}$ in $D_{i}$

  $max_{l}freq_{l,j}$ = frequency of occurrence of most frequently occurring term in $D_{j}$

  $f_{i,j} = \frac{freq_{i,j}}{max_{l}freq_{l,j}}$

- $f_{i,j}$ will replace $tf_{i,j}$ to calculate $w_{i,j}$
Example (using simple $w_{ij}$)

Assume $C$ is 2048 documents

Vocabulary $n$ is 3: oil, Mexico, refinery

Doc Frequency of terms is:

$$
\begin{align*}
DF_{\text{oil}} &= 128 \\
DF_{\text{mexico}} &= 16 \\
DF_{\text{refinery}} &= 1024
\end{align*}
$$

$$
 w_{ij} = tf_{ij} \times [\log_2(C) - \log_2(DF_i) + 1]
$$

Assume doc exists with following:

$$
\begin{align*}
TF_{\text{oil}} &= 4 \\
TF_{\text{Mexico}} &= 8 \\
TF_{\text{refinery}} &= 10
\end{align*}
$$
\[ w_{\text{oil}} = 4 \times (\log_2(2048) - \log_2(128) + 1) \]
\[ = 4 \times (11 - 7 + 1) = 20 \]

\[ w_{\text{Mexico}} = 8 \times (\log_2(2048) - \log_2(16) + 1) \]
\[ = 8 \times (11 - 4 + 1) = 64 \]

\[ w_{\text{refinery}} = 10 \times (\log_2(2048) - \log_2(1024) + 1) \]
\[ = 10 \times (11 - 10 + 1) = 20 \]
Benefits

• Can rank documents in order of relevance

• Can retrieve docs that partially match query

• Can use relevance feedback

Disadvantages

• Assumes term independence (but studies show that term dependence can hurt retrieval performance)

• No independent estimation of relevance: rank is dependent on other docs in collection

Probably most popular retrieval model after boolean