Query Operations

MURAD KAMALOV
Outline

- Introduction
- Feedback information from the user
- Expansion based on local set of documents
- Expansion based on global set of documents
- Conclusions
Introduction

- Increase number of relevant documents
- Decrease number of irrelevant documents
Introduction (cont.)

- Increase number of relevant documents
- Decrease number of irrelevant documents
- Ideal case
Increase number of relevant documents
Decrease number of irrelevant documents
Ideal case

Expansion of initial query
- add synonyms
- remove garbage terms

Example
initial = "a car"
expanded = "car vehicle automobile"
Initial query which we receive from user is never good enough to directly fetch the documents. It needs to be expanded.

In this presentation I will explain different methods, how expansion can be performed.
Vector Space Model

- Algebraic model to represent documents
- Vector representation of a document

$$v_d = [w_1, w_2, w_3, w_4, ...]$$

$$w_{t,d} = \frac{n_{t,d}}{\sum_k n_{k,d}} \cdot \log \frac{|N|}{|t \in d|}$$

- number of terms $$t$$ in document $$d$$
- number of all terms in document $$d$$
- number of documents in document set
- number of documents containing term $$t$$

$$w_i$$ - weight of a term $$i$$ in a document

term = word
Further...

- Introduction
- Feedback information from the user
- Expansion based on local set of documents
- Expansion based on global set of documents
- Conclusions
Feedback information from the user

1. User sends query Q to the search engine
2. Search engine retrieves all documents corresponding to query Q
3. Out of documents retrieved, user selects relevant ones and irrelevant ones and provides that information as a feedback to search engine
4. Based on this feedback search engine expands the query and returns improved search results
Feedback information from the user (cont.)

• **Query expansion - Vector Model**

\[
\vec{q}_{\text{expanded}} = a \cdot \vec{q}_{\text{original}} + \frac{b}{|D_r|} \cdot \sum_{\forall \vec{d}_j \in D_r} \vec{d}_j - \frac{c}{|D_n|} \cdot \sum_{\forall \vec{d}_j \in D_n} \vec{d}_j
\]

\[
a, b, c - \text{tuning constants}
\]

\[
D_r - \text{relevant documents returned by original query}
\]

\[
D_n - \text{irrelevant documents returned by original query}
\]

• **Query expansion - Probabilistic Model**

\[
\vec{q}_{\text{expanded}} = \sum_{i=1}^{t} w_{i,q} \cdot w_{i,D} \cdot \left( \log \frac{P(k_i \mid D_r)}{1 - P(k_i \mid D_r)} + \log \frac{1 - P(k_i \mid D_n)}{P(k_i \mid D_n)} \right)
\]

\[
w_{i,q} - \text{weight of i-th term in original query} \quad t - \text{number of terms in original query}
\]

\[
w_{i,D} - \text{weight of i-th term in set of all returned documents}
\]

\[
P(k_i \mid D_r) - \text{probability of finding term } k_i \text{ in relevant documents}
\]

\[
P(k_i \mid D_n) - \text{probability of finding term } k_i \text{ in irrelevant documents}
\]
Feedback information from the user (cont.)

- **Vector Model**
  - adds important terms to original query, based on relevant documents
  - removes unimportant terms from original query, based on irrelevant documents

- **Probabilistic model**
  - recalculates weights (importance) of query terms - some query terms are more important others less
  - doesn't add or remove terms

+ relevant/irrelevant documents are determined with high precision
- need user interaction to identify relevant/irrelevant documents
Expansion based on local set of documents

- Let's try to cluster relevant/irrelevant documents returned by original query automatically, without user intervention

*Local set of documents* = *documents retrieved by initial query*
Association Clustering

- Tries to find synonyms for query terms
  - terms that occur frequently in the same document are likely to be synonyms (or at least are related)

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>$f_{1,1}$</td>
<td>$f_{1,2}$</td>
<td>$f_{1,3}$</td>
</tr>
<tr>
<td>t2</td>
<td>$f_{2,1}$</td>
<td>$f_{2,2}$</td>
<td>$f_{2,3}$</td>
</tr>
<tr>
<td>t3</td>
<td>$f_{3,1}$</td>
<td>$f_{3,2}$</td>
<td>$f_{3,3}$</td>
</tr>
</tbody>
</table>

- Resulting matrix is called correlation matrix and it represents co-occurrence of terms in the same document.
- The higher is the correlation, more often terms occur in the same document.

```
<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>$f_{1,1}$</td>
<td>$f_{2,1}$</td>
<td>$f_{3,1}$</td>
</tr>
<tr>
<td>D2</td>
<td>$f_{1,2}$</td>
<td>$f_{2,2}$</td>
<td>$f_{2,2}$</td>
</tr>
<tr>
<td>D3</td>
<td>$f_{1,3}$</td>
<td>$f_{2,3}$</td>
<td>$f_{3,3}$</td>
</tr>
</tbody>
</table>
```

```
m * m^T =
```

```
<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t1*t1</td>
<td>t1*t2</td>
<td>t1*t3</td>
</tr>
<tr>
<td>t2</td>
<td>t2*t1</td>
<td>t2*t2</td>
<td>t3*t2</td>
</tr>
<tr>
<td>t3</td>
<td>t3*t1</td>
<td>t3*t2</td>
<td>t3*t3</td>
</tr>
</tbody>
</table>
```
Association Clustering: Example

If $t3*t1$ and $t3*t2$ are sufficiently large, it means that terms $t3$, $t2$, $t1$ occur frequently together in the documents and thus form association cluster around $t3$.

Hence query which contains term $t3$ can be expanded with terms $t2$ and $t1$.

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>t1*t1</td>
<td>t1*t2</td>
<td>t1*t3</td>
</tr>
<tr>
<td>t2</td>
<td>t2*t1</td>
<td>t2*t2</td>
<td>t3*t2</td>
</tr>
<tr>
<td>t3</td>
<td>t3*t1</td>
<td>t3*t2</td>
<td>t3*t3</td>
</tr>
</tbody>
</table>
Metric Clustering

• Metric clustering considers distance between terms

• Terms close to each other, in the text, are more likely to be related, than terms farther from each other.

• Correlation matrix contains average distances of terms from each other based on local document set.
Expansion based on local set of documents (cont.)

- Allows to automatically expand query based on documents retrieved from the initial query
- Can create a lot of overhead, because for each query correlation matrices should be recalculated
- Hence method is unsuitable for systems where query response times should be small.
Expansion based on global set of documents

- Similar to expansion using local set of documents with association clustering.
- Correlation matrix is computed for all the documents in system.
- Resulting structure is called Thesaurus.
- Creating such huge structure is computationally expensive, but it should be done only once and later it can be incrementally updated.

Global set of documents = all the documents in the system
Approach based on global set of documents (cont.)

Query processing is performed as follows:

- Query is represented as terms:
  \[ q = (t_1, t_2, t_3, \ldots) \]

- Based on thesaurus, find all the document terms similar to query terms:
  \[ (k_1, k_2, k_3, \ldots) \text{ are similar to terms } (t_1, t_2, t_3, \ldots) \]

- For each similar term we compute how similar it is to the whole query:
  \[ \text{Similarity of } (k_1, k_2, k_3, \ldots) \text{ to whole } q \]

- Expand the original query with terms which are the most similar to the original query.
Approach based on global set of documents (cont.)

- No need to query for documents twice
- Suitable for systems where small response times are required
- Improves retrieval performance in range of 20%
Conclusions

- Techniques of expanding the query for improving relevancy of search engine responses.

- No silver bullet, each approach is good for its own purpose
  - User feedback based
  - Local document set based
  - Global document set based
References

- Modern Information Retrieval, Chapter 5, Query Operations, book by Ricardo Baeza-Yates and Berthier Ribeiro-Neto
Thank You!