Models for Document & Query Representation

Ziawasch Abedjan
Overview

- Introduction & Definition
- Boolean retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Overview

- Introduction & Definition
- Boolean retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Introduction

**Information Retrieval:**

*Finding material of an unstructured nature that satisfies an information need from within large collections*

- Application areas:
  - Personal IR (junk mail filter, grep)
  - Enterprise, institutional, and domain-specific search
  - Web search (Search over billions of documents)
- Information request is represented as a query
- Challenges:
  - Process large document collections quickly
  - More flexible matching
  - Ranked retrieval

Advanced retrieval models are needed!
Overview

- Introduction & Definition
- Boolean Retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Boolean Retrieval Model

For each document in the corpus it is recorded whether it contains each word out of all words in the corpus or not

- Documents are sets of words
- Term-document matrix:

<table>
<thead>
<tr>
<th></th>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othelo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Relies on the use boolean operators
- Terms in queries are linked with AND, OR, and NOT
  - I.e.: Brutus AND Caesar AND NOT Calpurnia
    
    \[
    11010 \text{ AND } 11011 \text{ AND } 11011 = 11010
    \]
**Problem:** A term-document matrix is too big in a more realistic scenario

**Observation:** Matrix is very sparse

**Solution:**

- Record only the 1 positions
- Use an inverted index and a dictionary of terms
- For each term there is a sorted postings list ad the document frequency
- Posting: docID

<table>
<thead>
<tr>
<th>Term</th>
<th>Doc.freq.</th>
<th>Postings list</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brutus</td>
<td>3</td>
<td>1,2,4</td>
</tr>
<tr>
<td>Caesar</td>
<td>4</td>
<td>1,2,4,5</td>
</tr>
</tbody>
</table>
Boolean Retrieval Model 3

Query processing:
i.e. Brutus AND Caesar
(1) Locate Brutus in the Dictionary
(2) Retrieve its postings
(3) Locate Caesar in the Dictionary
(4) Retrieve its postings
(5) Intersect the two postings lists

- Intersection algorithm is linear in the total number of postings entries $O(N)$, if the postings are sorted globally
- **Query optimization:**
  - Access shorter postings lists first
  - Look up document frequency from the dictionary
BRM Boundaries

- Precise query language with operators is needed, free text queries would be better
  - i.e. google query

- Tolerance to spelling mistakes and inconsistent choice of words

- Proximity queries
  - i.e. **Gates** NEAR **Microsoft**

- Retrieval should consider term frequency within documents

- Ranking the returned Results is not possible by the boolean Model
Ranked boolean retrieval:
- Scores have to be computed for each document
- Idea:
  - Weighting the importance of terms in documents
  - Use statistics of the terms for the weights

Parametric indexes:
- Documents consist of different zones (body, title, other meta data)
- Different zones have different weights $g_z$
Example

Weights for body and title:

\[ g_b = 0.3, \quad g_t = 0.7, \]
\[ g_b + g_t = 1 \]

Scoring function:

\[ score(d, q) = g_t \cdot contains_t(d, q) + g_b \cdot contains_b(d, q) \]

Query: Caesar

<table>
<thead>
<tr>
<th>Title</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Julius Caesar</td>
<td>0.7<em>1 + 0.3</em>1</td>
</tr>
<tr>
<td>Hamlet</td>
<td>0.7<em>0 + 0.3</em>1</td>
</tr>
</tbody>
</table>
Overview

- Introduction & Definition
- Boolean Retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Vector Space Model

- Set of documents as vectors in a common vector space with one axis for each term
  - Vector for document \( \vec{V}(d) \)
  - Queries are vectors in the same vector space
- Similarity of two vectors defined by cosine similarity

\[
sim(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{||\vec{V}(d_1)|| \cdot ||\vec{V}(d_2)||}
\]
How to weight terms?

Weighting terms (vector components):

- Term frequency \( tf_{t,d} \) indicates how many times a term \( t \) occurs in a document \( d \).
- Document frequency \( df_t \) indicates in how many documents a term occurs.
- Inverse document frequency \( idf_t = \log\frac{N}{df_t} \), where \( N \) is the total number of documents.

\[
\text{tf} - \text{idf} \text{ Weighting: } \quad tf - idf_{t,d} = tf_{t,d} \times idf_t
\]

Derived scoring function:

\[
\text{score}(d,q) = \frac{\sum_{t \in q} tf - idf_{t,d}}{|V(d)|}
\]
Score for the document: 
0 + 0.82 + 2.46 = 3.28

\[
score(d,q) = \frac{\sum_{i \in q} tf - idf_{i,d}}{|V(d)|}
\]
Overview

- Introduction & Definition
- Boolean Retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Probabilistic Information Retrieval

- Boolean and vector space models:
  - Formally defined but semantically imprecise calculus of index terms
  - Uncertain guess of relevance of documents to the information need
- Probability theory provides a foundation for such reasoning
  - Estimate how likely it is that a document is relevant to an information need
- Assumption: We know that some documents are relevant
  - Based on statistics and relevance feedback
Binary Independent Model

- Random variable $R_{d,q}$ indicates whether $d$ is relevant to a given query $q$
- Rank documents by their estimated probability of relevance:

$$P(R = 1 \mid d, q)$$

- Binary:
  - Documents and queries are both represented as binary term incidence vectors
- Independence:
  - Terms occur in documents independently
BIM Relevance Feedback

Feedback loop:
1. Initial estimates of relevance (i.e. 0.5)
2. Retrieve a set of candidate documents
3. Interact with user to refine the set of relevant documents
4. Reestimate the relevance probability
5. Repeat the above process from step 2, until the user is satisfied
Overview

- Introduction & Definition
- Boolean Retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Language Model Approach

- Idea:
  A document is a good match for a query if the document model is likely to generate the query

- Conditions to be met:
  - Accurate Representation of the data
  - Approach should be understandable to users
  - Users should get some sense of term distribution

- Each document $d$ has its own probabilistic language model $M_d$

- Rank documents by the probability $P(q \mid M_d)$
Language Models

- Nondeterministic finite automata:

- Language model:

  A function that puts probability measures over strings drawn from some vocabulary.

  \[ \sum_{t \in \mathcal{V}} P(t) = 1 \]
Types of Models

Unigram language model:

\[ P(t_1, t_2, t_3) = P(t_1)P(t_2)P(t_3) \]

Bigram language model:

\[ P(t_1, t_2, t_3) = P(t_1)P(t_2 \mid t_1)P(t_3 \mid t_2) \]

Most IR models use unigram language models.
Example

<table>
<thead>
<tr>
<th>Term</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>the</td>
<td>0.2</td>
</tr>
<tr>
<td>a</td>
<td>0.1</td>
</tr>
<tr>
<td>frog</td>
<td>0.01</td>
</tr>
<tr>
<td>toad</td>
<td>0.01</td>
</tr>
<tr>
<td>said</td>
<td>0.03</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[
P(frog \text{ said toad} \mid M_1) = (0.01 \times 0.03 \times 0.01) = 0.000003
\]

\[
P(frog \text{ said toad} \mid M_2) = (0.0002 \times 0.03 \times 0.0001) = 0.0000000006
\]

\[
P(frog \text{ said toad} \mid M_1) > P(frog \text{ said toad} \mid M_2)
\]
Query Likelihood Model

Rank documents by $P(d \mid q)$ where the probability of a document is interpreted as the likelihood that it is relevant to the query.

$$P(d \mid q) = \frac{P(q \mid d)P(d)}{P(q)}$$

$P(d), P(q)$ Can be both ignored

$$P(d \mid q) = P(q \mid d) = P(q \mid M_d)$$

Estimation:

$$\hat{P}(q \mid M_d) = \prod_{t \in q} \hat{P}(t \mid M_d) = \prod_{t \in q} \frac{tf_{t,d}}{L_d}$$
QLM Smoothing

\[
\hat{P}(q \mid M_d) = \prod_{t \in q} \hat{P}(t \mid M_d) = \prod_{t \in q} \frac{tf_{t,d}}{L_d}
\]

Zero probability is a problem! Smoothing is needed.

Idea:
- If a term is not generated by a LM of a document then
  \[
  \hat{P}(t \mid M_d) \leq \frac{cf_t}{T}
  \]
  - \( cf_t \) raw count of the term
  - \( T \) number of tokens in the corpus

Information Retrieval | Ziawasch Abedjan | 09 December 2008
QLM Smoothing

\[ P(d \mid q) = \hat{P}(q \mid M_d) = \prod_{t \in q} ((1 - \lambda) \hat{P}(t \mid M_c) + \lambda \hat{P}(t \mid M_d)) \]

Linear interpolation with \(0 < \lambda < 1\) and \(M_c\) is a language model built from the entire corpus.
## Results of a comparison of tf-idf with LM term weighting by Ponte and Croft (1998)

<table>
<thead>
<tr>
<th>Recall</th>
<th>tf-idf</th>
<th>LM</th>
<th>% Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.7439</td>
<td>0.7590</td>
<td>+2.0</td>
</tr>
<tr>
<td>0.1</td>
<td>0.4521</td>
<td>0.4910</td>
<td>+8.6</td>
</tr>
<tr>
<td>0.2</td>
<td>0.3514</td>
<td>0.4045</td>
<td>+15.1</td>
</tr>
<tr>
<td>0.3</td>
<td>0.2761</td>
<td>0.3342</td>
<td>+21.0</td>
</tr>
<tr>
<td>0.4</td>
<td>0.2093</td>
<td>0.2572</td>
<td>+32.3</td>
</tr>
<tr>
<td>0.5</td>
<td>0.1558</td>
<td>0.2061</td>
<td>+32.3</td>
</tr>
<tr>
<td>0.6</td>
<td>0.1024</td>
<td>0.1405</td>
<td>+37.1</td>
</tr>
<tr>
<td>0.7</td>
<td>0.0451</td>
<td>0.0760</td>
<td>+68.7</td>
</tr>
<tr>
<td>0.8</td>
<td>0.0160</td>
<td>0.0432</td>
<td>+169.6</td>
</tr>
<tr>
<td>0.9</td>
<td>0.0033</td>
<td>0.0063</td>
<td>+89.3</td>
</tr>
<tr>
<td>1.0</td>
<td>0.0028</td>
<td>0.0050</td>
<td>+76.9</td>
</tr>
<tr>
<td><strong>Ave</strong></td>
<td><strong>0.1868</strong></td>
<td><strong>0.2233</strong></td>
<td><strong>+19.55</strong></td>
</tr>
</tbody>
</table>
Extended LM Approaches

- Query likelihood model (a)
- Document likelihood model (b)
- Model comparison (c)
Overview

- Introduction & Definition
- Boolean Retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary
Summary

- **Boolean retrieval model**
  - Can detect whether a term occurs in a document or not
  - Depends on boolean operators
  - Boolean ranked retrieval bases still does not consider tf and df

- **Vector space model**
  - Document scores depend on statistics like tf, df and idf
  - Semantic of terms is still not considered

- **Probabilistic information retrieval**
  - Document scores are computed by probabilistic estimations
  - Considers semantic by referring to relevance feedback

- **Language model approach**
  - Based on probabilistic language modeling
  - Conceptually simple and explanatory
References

- [www.informationretrieval.org](http://www.informationretrieval.org) (Chapters 1, 6, 11, 12)
- "A Language Modeling Approach to Information Retrieval" (Ponte and Croft)
Overview

- Introduction & Definition
- Boolean Retrieval
- Vector Space Model
- Probabilistic Information Retrieval
- Language Model Approach
- Summary

Any questions?