Retrieval Models: Ranked Boolean

Advantages:
• All advantages from unranked Boolean algorithm
  – Works well when query is precise; predictive; efficient
• Results in a ranked list (not a full list); easier to browse and find the most relevant ones than Boolean
• Rank criterion is flexible: e.g., different variants of term evidence

Disadvantages:
• Still an exact match (document selection) model: inverse correlation for recall and precision of strict and loose queries
• Predictability makes user overestimate retrieval quality
Retrieval Models: Vector Space Model

- Any text object can be represented by a term vector
  - Documents, queries, passages, sentences
  - A query can be seen as a short document
- Similarity is determined by distance in the vector space
  - Example: cosine of the angle between two vectors

(Research) Famous Examples
- The SMART system
  - Developed at Cornell University: 1960-1999
  - Still quite popular
- The Lucene system
  - Open source information retrieval library; (Based on Java)
  - Works with Hadoop (Map/Reduce) in large scale app (e.g., Amazon Book)

Retrieval Models: Vector Space Model

Vector space model vs. Boolean model

- Boolean models
  - Query: a Boolean expression that a document must satisfy
  - Retrieval: Deductive inference
- Vector space model
  - Query: viewed as a short document in a vector space
  - Retrieval: Similarity search
Retrieval Models: Vector Space Model

- Vector representation

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Oracle</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Starbucks</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

© 2019 Christopher W. Clifton
Retrieval Models: Vector Space Model

Give two vectors of query and document
- query \( \vec{q} = (q_1, q_2, \ldots, q_n) \)
- document \( \vec{d}_j = (d_{j_1}, d_{j_2}, \ldots, d_{j_m}) \)
- calculate the similarity

Cosine similarity: Angle between vectors

\[
\text{sim}(\vec{q}, \vec{d}_j) = \cos(\theta(\vec{q}, \vec{d}_j))
\]

\[
\cos \left( \theta \left( \vec{q}, \vec{d}_j \right) \right) = \frac{\vec{q} \cdot \vec{d}_j}{\|\vec{q}\| \|\vec{d}_j\|} = \frac{q_1d_{j_1} + q_2d_{j_2} + \ldots + q_nd_{j_n}}{\sqrt{q_1^2 + \ldots + q_n^2} \sqrt{d_{j_1}^2 + \ldots + d_{j_n}^2}}
\]

Retrieval Models: Vector Space Model

- Vector representation

<table>
<thead>
<tr>
<th></th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Oracle</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>Starbucks</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Similarity Score</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query</td>
<td>0.59</td>
<td>0.99</td>
<td>0.70</td>
</tr>
</tbody>
</table>
Retrieval Models:
Vector Space Model

Vector Coefficients

- The coefficients (vector elements) represent term evidence/term importance
- Derived from several elements
  - Document term weight: Evidence of the term in the document/query
  - Collection term weight: Importance of term from observation of collection
  - Length normalization: Reduce document length bias
- Naming convention for coefficients:

\[ q_k . d_{j,k} = DCL . DCL \]

First triple represents query term; second for document term

Retrieval Models:
Vector Space Model

- Common vector weight components:
- Inc.ltc: widely used term weight
  - “l”: \( \log(tf) + 1 \)
    - 0 if \( tf = 0 \)
  - “n”: no weight/normalization
  - “t”: \( \log(N/df) \)
  - “c”: cosine normalization

\[
\frac{q_{d,m} + q_{d,j} + \ldots + q_{d,m}}{\sqrt{\sum_{k} \left( \log(tf_{q,k}) + 1 \right)^2}} \cdot \frac{\sqrt{\sum_{k} \left( \log(tf_{d,k}) + 1 \right)^2}}{\sqrt{\sum_{k} \left( \log(tf_{d,k}) + 1 \right) \log \frac{N}{df_{d,k}}}}
\]
Retrieval Models: Vector Space Model

- Common vector weight components:
  - dnn.dtb: handle varied document lengths
    - “d”: 1+ln(1+ln(tf))
    - “t”: log((N/df))
    - “b”: 1/(0.8+0.2*doclen/avg_doclen)

Retrieval Models: Vector Space Model Summary

- Standard vector space
  - Represent query/documents in a vector space
  - Each dimension corresponds to a term in the vocabulary
  - Use a combination of components to represent the term evidence in both query and document
  - Use similarity function to estimate the relationship between query/documents (e.g., cosine similarity)
Retrieval Models: Vector Space Model

Advantages:
- Best match method; it does not need a precise query
- Generates ranked lists; easy to explore the results
- Simplicity: easy to implement
- Effectiveness: often works well
- Flexibility: can utilize different types of term weighting methods
- Used in a wide range of IR tasks: retrieval, classification, summarization, content-based filtering...

Retrieval Models: Vector Space Model

- “l”: \( \log(tf+1) \)
- “n”: no weight/normalization
- “t”: \( \log(N/df) \)
- “c”: cosine normalization

\[
\frac{q_1d_{11} + q_2d_{12} + \cdots + q_md_{1m}}{\|q_d\|} = \frac{\sum_k \left[ \log(tf,(k) + 1) \right] \left[ \log(tf,(k) + 1) \log \frac{N}{df,(k)} \right]}{\sqrt{\sum_k \left[ \log(tf,(k) + 1) \right]^2} \sqrt{\sum_k \left[ \log(tf,(k) + 1) \log \frac{N}{df,(k)} \right]^2}}
\]
Retrieval Models: Vector Space Model

Disadvantages:

- **Hard to choose the dimension of the vector ("basic concept")**
  - Terms may not be the best choice
- Assume independent relationship among terms
- Heuristic for choosing vector operations
  - Choose of term weights
  - Choose of similarity function
- Assume a query and a document can be treated in the same way

What is a good vector representation?

- **Orthogonal**: the dimensions are linearly independent ("no overlapping")
- No ambiguity (e.g., Java)
- Wide coverage and good granularity
- Good interpretation (e.g., representation of semantic meaning)
- Many possibilities: words, stemmed words, "latent concepts"...