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

### 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>
Retrieval Models: Vector Space Model

- Vector representation

![Diagram showing vector space model with query and document vectors](image)

Give two vectors of query and document

- query \( \vec{q} = (q_1, q_2, ..., q_n) \)
- document \( d_j = (d_{j1}, d_{j2}, ..., d_{jn}) \)
- calculate the similarity

Cosine similarity: Angle between vectors

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

\[
\cos \left( \theta \left( \vec{q}, \vec{d}_j \right) \right) = \frac{\vec{q} \cdot \vec{d}_j}{\|\vec{q}\| \|\vec{d}_j\|} = \frac{q_1 d_{j1} + q_2 d_{j2} + ... + q_n d_{jn}}{\sqrt{q_1^2 + q_2^2 + ... + q_n^2} \sqrt{d_{j1}^2 + d_{j2}^2 + ... + d_{jn}^2}}
\]
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>

Similarity Score

<table>
<thead>
<tr>
<th></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:

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

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
      \[
      \frac{q_d \cdot d_k + q_d \cdot d_j + \cdots + q_d \cdot d_m}{\|q_d\| \cdot \|d\|} = \frac{\sum_k \left[ \log(g_f(k) + 1) \cdot \log(g_f(k) + 1) \cdot \log N \right]}{\sum_k \left[ \log(g_f(k) + 1) \right] \cdot \sum_k \left[ \log(g_f(k) + 1) \cdot \log N \right]}
      \]

- dnn.dtb: handle varied document lengths
  - “d”: 1+ln(1+ln(tf))
  - “t”: log((N/df))
  - “b”: 1/(0.8+0.2*docleng/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…