Evaluating Relevancy Of Words In Document Queries Using Vector Space Model

Deepika Matta, M-tech Scholar, Department of Computer Science, RPIIT, Karnal, India
Manoj Verma, HOD, Department of Computer Science RPIIT Karnal, India

ABSTRACT
In this paper we evaluate result of applying vector space model to determine the relevancy of words in a document queries, in vector space model we applying TF/IDF technique for calculating values of each word and we conclude that the query which appears most relevant to document queries seems first on the web page

Keywords: VSM(vector space model), TF/IDF (term frequency/inverse frequency document), CAD (cosine angle Distance).

Query Retrieval Problem
The task of retrieving data from a user defined query has become so common and natural in recent years that some might not give it a second thought however, this growing use of query retrieval warrants continued research and enhancements to generate better solutions to the problem

Informally, query retrieval can be described as the task of searching a collection of data is that text documents databases, networks, etc for specific instances of that data. First we will limit ourselves to searching a collection of English documents. The refund problem then becomes the task of searching this corpus for documents that the query retrieval system considers relevant to what the user entered as the query.

Let us describe this problem more formally we have a set of documents D, with the user entering a query q=w1, w2……wn for a sequence of words wi. Then we wish to return a subset D* of D such that for each d belongs to D, we maximize the following probability: P(d|q,d).

As the above notation suggests, numerous approaches to this problem involve probability and statistics, while others propose vector based models to enhance the retrieval(1).

Application of Information Retrieval: Search Engine
The key element is not the data it self .it is the engine that extract interesting data and executes queries upon it, the requirement for such an engine are very high. The final product must posses both, good performance and a good rating, designing this engine was the main challenge and required some research in the field of information retrieval. it is useful to take an look on available search engine to get an idea how solutions looks like.(6)

Vector Space Model
Vector space model is an algebraic model for representing text documents (and any objects, in general) as vectors of identifiers, such as, for example, index terms. It is used in information filtering, information retrieval, indexing and relevancy rankings.

In order to reduce the complexity of the documents and make them easier to handle, the document has to be transformed from the full text version to a document vector which describes the contents of the document.

TF/IDF Technique
Term Frequency/Inverse Document Frequency model is based on the principle that if a term occurs more within a document(TF) and rare within the corpus of documents(IDF), then that term would be having high discriminative power to distinguish between relevant and non-relevant documents[3].The document having high TF/IDF value is having strong relationship with the query and would be more relevant for the user.

Inverse Document Frequency (IDF) is based on the fact that a term which occurs in many documents is not a good discriminator and should be given less weight than one which occurs in few documents [4]. If there are N documents in the collection, and that term ti occurs in ni of them. IDF is calculated as

\[
\text{idf}(t_i) = \log \frac{N}{n_i}
\]
Term Frequency/Inverse Document Frequency model incorporates local and global information. Encoding TF-IDF is simple. Term Weight is calculated as

\[ w_i = \text{tf}_i \times \log \left( \frac{D}{\text{df}_i} \right) \]

where,

a) \( \text{tf}_i \) = term frequency (term counts) or number of times a term i occurs in a document. This accounts for local information.

b) \( \text{df}_i \) = document frequency or number of documents containing term i

c) \( D \) = number of documents in a database.

### Example Based On Vector Space Model

Let us assume we deal with a basic term vector model in which we

1. Do not take into account WHERE the term occur in document
2. Use all term, including very common term and stop words.
3. Do not reduce terms to root terms.
4. Use raw frequencies for terms queries

Suppose we query an IR system for the query “diamond is better”

The database collection consists of three documents (D=3) with the following content:

- D1: "Shipments of gold damaged in a fire"
- D2: "Delivery of silver arrived in a silver truck"
- D3: "Shipments of gold arrived in a truck"

#### Term Vector Model Based on \( w_i = \text{tf}_i \times \text{IDF}_i \)

<table>
<thead>
<tr>
<th>Terms</th>
<th>Counts, tf</th>
<th>D1</th>
<th>D2</th>
<th>D3</th>
<th>D/dfi</th>
<th>IDF_i</th>
<th>Weights, ( w_i = \text{tf}_i \times \text{IDF}_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0 1 1 1 3</td>
<td>3/3 = 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>arrived</td>
<td>0 0 1 1 2</td>
<td>3/2 = 1.5</td>
<td>0.176'</td>
<td>0</td>
<td>0</td>
<td>0.1761</td>
<td>0.1761</td>
</tr>
<tr>
<td>damaged</td>
<td>0 1 0 0 1</td>
<td>3/1 = 3</td>
<td>0.477'</td>
<td>0</td>
<td>0.4771</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>delivery</td>
<td>0 0 1 0 1</td>
<td>3/1 = 3</td>
<td>0.477'</td>
<td>0</td>
<td>0.4771</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fire</td>
<td>0 1 0 0 1</td>
<td>3/1 = 3</td>
<td>0.477'</td>
<td>0</td>
<td>0.4771</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>gold</td>
<td>1 1 0 1 2</td>
<td>3/2 = 1.5</td>
<td>0.176'</td>
<td>0.1761</td>
<td>0</td>
<td>0</td>
<td>0.1761</td>
</tr>
<tr>
<td>in</td>
<td>0 1 1 1 3</td>
<td>3/3 = 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>of</td>
<td>0 1 1 1 3</td>
<td>3/3 = 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>silver</td>
<td>1 0 2 0 1</td>
<td>3/1 = 3</td>
<td>0.477'</td>
<td>0.4771</td>
<td>0</td>
<td>0</td>
<td>0.9542</td>
</tr>
<tr>
<td>shipment</td>
<td>0 1 0 1 2</td>
<td>3/2 = 1.5</td>
<td>0.176'</td>
<td>0</td>
<td>0.1761</td>
<td>0</td>
<td>0.1761</td>
</tr>
<tr>
<td>truck</td>
<td>1 0 1 1 2</td>
<td>3/2 = 1.5</td>
<td>0.176'</td>
<td>0.1761</td>
<td>0</td>
<td>0</td>
<td>0.1761</td>
</tr>
</tbody>
</table>

1. Columns 1-5, first, we construct an index of terms from the documents and determine the term counts \( \text{tf}_i \) for the query and each document \( D_j \).
2. Columns 6-8 second we compute the document frequency \( \text{df}_i \) for each document since \( \text{IDF}_i = \log(D/\text{df}_i) \) and \( D=3 \), this calculation is straightforward.
3. Columns 9-12 third, we take the \( \text{tf} \times \text{IDF} \) products and compute the term weights. These columns can be viewed as a sparse matrix in which most entries are zero(5).
Similarity Analysis:
First for each document and query we compute all vector lengths (zero terms ignored)

\[ |Dj| = \sqrt{0.4771^2 + 0.4771^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.5173} = 0.7192 \]
\[ |D2| = \sqrt{0.1761^2 + 0.4771^2 + 0.0954^2 + 0.1761^2} = \sqrt{1.2001} = 1.0955 \]
\[ |D3| = \sqrt{0.1761^2 + 0.1761^2 + 0.1761^2 + 0.1761^2} = \sqrt{0.1240} = 0.3522 \]
\[ |Q| = \sqrt{0.1761^2 + 0.4771^2 + 0.761^2} = \sqrt{0.2896} = 0.5382 \]
\[ |Dj| = \sqrt{\sum w_{i,j}^2} \]
\[ |Q| = \sqrt{\sum w_{Q,j}^2} \]

Next we compute all dot products (zero products ignored). When mathematicians want to multiply two vectors together, one way they do it is with an operator called a “dot product”. Its symbol is a big huge fat dot “.” the result of a dot product operation is a scalar, not a vector (2).

\[ Q \cdot D1 = 0.1761 \cdot 0.1761 = 0.0310 \]
\[ Q \cdot D2 = 0.4771 \cdot 0.9542 + 0.1761 \cdot 0.1761 = 0.4862 \]
\[ Q \cdot D3 = 0.1761 \cdot 0.1761 + 0.1761 \cdot 0.1761 = 0.0620 \]
\[ Q \cdot D1 = \sum w_{Q,j} w_{i,j} \]

Now we compute similarity values

\[ \cosine \theta_{D1} = \frac{Q \cdot D1}{|Q| \cdot |D1|} = \frac{0.0310}{0.5382 \cdot 0.7192} = 0.0801 \]
\[ \cosine \theta_{D2} = \frac{Q \cdot D2}{|Q| \cdot |D2|} = \frac{0.4862}{0.5382 \cdot 1.0955} = 0.4246 \]
\[ \cosine \theta_{D3} = \frac{Q \cdot D3}{|Q| \cdot |D3|} = \frac{0.0620}{0.5382 \cdot 0.3522} = 0.3271 \]
\[ \cosine \theta_{D1} = \text{Sim}(Q, D1) \]
\[ \text{Sim}(Q, D1) = \frac{\sum w_{Q,j} w_{i,j}}{\sqrt{\sum w_{Q,j}^2} \sqrt{\sum w_{i,j}^2}} \]

Finally, the documents are sorted and ranked in descending order according to the similarity values:
- Rank 1: Doc 2 = 0.8246
- Rank 2: Doc 3 = 0.3271
- Rank 3: Doc 1 = 0.0801

Limitations:
1. Long documents
2. False negative matches.
3. Semantic content. (Need to use special tags).

We can improve this model by
1. Getting a set of keywords that are representative of each document.
2. Eliminating all stop words.

Conclusion
Using vector space model and TF-IDF technique we conclude that the query which has high relevancy appears first when a user put some query on the web.

References
[2] 2. The classic vector space model Dr.E. Garcia, 06.