Vector Space Model - IR weekly report 6

hello underworld

14th August 2021

1 Introduction

Common Boolean Retrieval could result some problems possibly due to whether the person is expert in boolean retrievaling. Boolean queries often result in either too few (=0) or too many (1000s) results. As a result, we introduce the ranked retrieval to return in order the results most useful to users. To this end, we have to take a model to compute the score of each document in the collections due to the query to rank these target documents.

2 Some Initial Concepts

2.1 Term Frequency(TF)

The times a term in the query occurs in a document or a zone is the tf of the term in the document. The reason why we introduce this TF is that a document or zone that mentions a query term more often has more to do with that query and therefore should receive a higher score.

2.2 Bag of Words Model

To simplify the model that computing the score of each document, we implement the bag of words model, the exact ordering of the terms in a document is ignored but the number of occurrences of each term is material (in contrast to Boolean retrieval). We only retain information on the number of occurrences of each term. Thus, the document 'Mary is quicker than John'is, in this view, identical to the document 'John is quicker than Mary'.

2.3 Inverse Document Frequency(IDF)

There is still a question that different terms in a document should not be equally important like the *stop words* playing a tiny role in this model although the times they occurs a lot.Rare terms are more informative than frequent terms, so, we introduce the idf and df. df is the document frequency of t: the number of documents that contain t. Then, We define the idf (inverse document frequency) of t by this expression:

$$idf = log_{10}(N/df)$$

In this expression, we could know that, when a term occurs in less documents, the idf of it will be higher and vice versa.

3 Vector Space Model

3.1 tf-idf weighting

Now, we could compute the weight of each term by this expression following:

$$W_t = df_t * idf_t$$

Finally,we could define the score for matching the query and a document by this expression:

$$score_{q,d} = \Sigma_{t \in q \cap d} tf_t * idf_t$$

When the query is given, we could compute the weight of each term in the query for the document.However, due to the bag of words model, the query is a dictionary containing some terms, ignoring the relationship between terms.

3.2 Vector Space Model(VSM)

So, here, we introduce the VSM model. In this model, each term is the axes of the space, while docuemnts are the points or vectors. The value of the document on some dimension is the weight of the corresponding term (calculated by the tf-idf). And, finally, a vector or point is representing a document in this space.

Now, through this model, we could get the target documents which are high-ranked in proximity with the query, by calculating the similarity of vectors representing the query and the documents.

3.3 Calculating Similarity

How do we quantify the similarity between two documents in this vector space? A first attempt might consider the magnitude of the vector difference between two document vectors. This measure suffers from a drawback: two documents with very similar content can have a significant vector difference simply because one is much longer than the other. Thus the relative distributions of terms may be identical in the two documents, but the absolute term frequencies of one may be far larger.

To compensate for the effect of document length, the standard way of quantifying the similarity between two documents d1 and d2 is to compute the cosine similarity of their vector representations by this following expression.

$$sim(d_1, d_2) = \vec{V}(d_1) * \vec{V}(d_2) \ / \ (|\vec{V}(d_1)| * |\vec{V}(d_2)|)$$

In this way, we can replace the d_1 with the target query, and then, calculating the proximity between them. Here is an example for the idea:

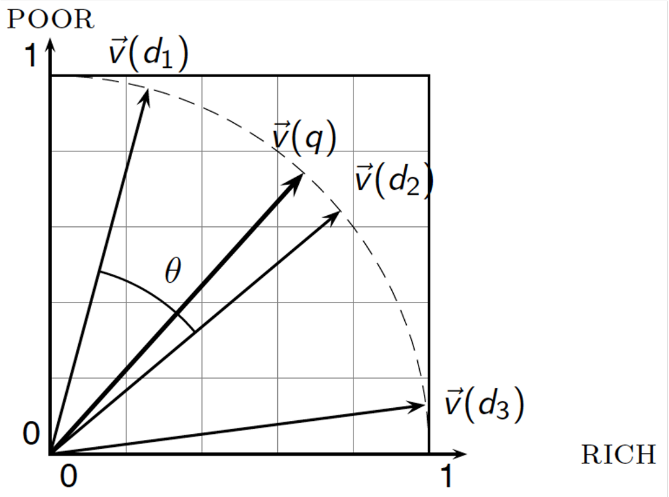

Figure 1: cosine similarity