Introduction to Search and Recommender Systems

Chirag Shah

Classical IR: Vector Space Models, Probabilistic Models, Language Models

Reading List:

- Chapter 2 - Baeza-Yates, R. (Ricardo)., Ribeiro-Neto, B. Modern information retrieval. New York: ACM Press.
- ChengXiang Zhai. Statistical Language Models for Information Retrieval
- Victor Lavrenko and W. Bruce Croft. 2017. Relevance-Based Language Models. SIGIR Forum, 51, 2 (July 2017), 260–267. DOI:https://doi.org/10.1145/3130348.3130376

Vector Space Models

After realizing that binary weights are too limiting for real practices, the Vector Model is developed to assign non-binary weights to indexes in queries and documents. The weights are used to compute the degree of similarity between each document and the query term. Then the resulting list of documents is gained by ranking the retrieved documents in descending order of degree of similarity.

There are actually several models of retrieval, but none is as popular as the vector space model in IR. And it is not only in IR, but several other fields where this model is widely used. The basic concept of the model is really simple. It is based on a key idea that a piece of information (document or query) is a vector in high-dimensional space.

Let us understand this by a toy example. Imagine our vocabulary is {dog,cat}, mere two words. This means we are working on a two-dimensional space when it comes to the vector space model. These two dimensions are shown in Figure 1. Now, let us assume we have one document with five occurrences of 'dog', two occurrences of 'cat', and nothing else. In the space given, this vector {5,2} is represented in Figure 1 as Vd. And let us assume that we have a query with one occurrence of 'dog' and one occurrence of 'cat. This vector {1, 1} is represented in Figure 1 as Vq.

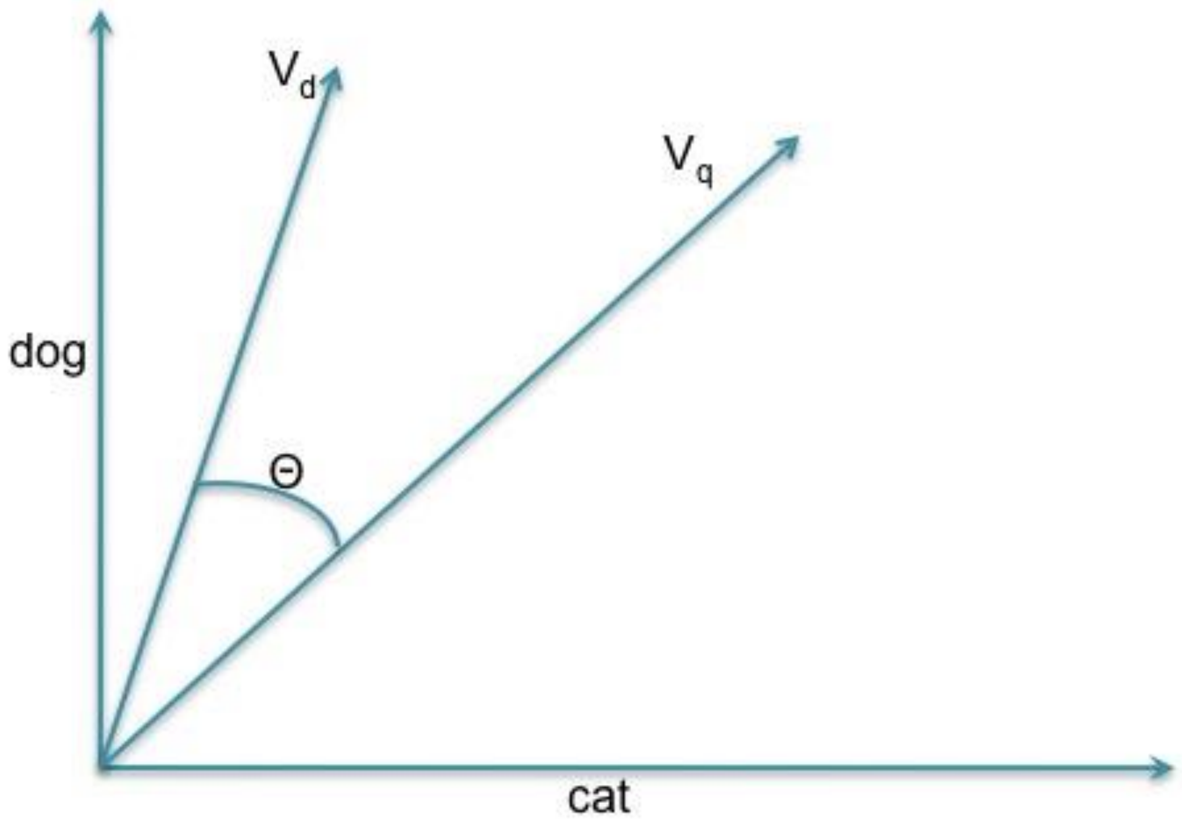

Figure 1: A two-dimensional vector space.

In order to evaluate the matching of these two vectors, we need to see how “close” they are in the given space. One way of doing this is by looking at the angle θ between these two vectors. This brings us to the most common way of finding similarity among two vectors in the vector space model – cosine similarity, or dot product of two vectors.

$$\begin{aligned} sim(d_j, q) &= \frac{\vec{d_j} \bullet \vec{q}}{|\vec{d_j}| \times |\vec{q}|} \\ &= \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}} \end{aligned}$$

* w_i,j, w_i,q > 0 (non-binary) is the weight associated with index term k_i of a document d_j
* d_j=document and q=user query are two vectors in dimension t, where t is the number of index terms

Plugging in the values Vd = {5, 2} and Vq = {1, 1} in this equation, we get 0.92 as the similarity score. Since cosine scores range from 0 to 1, we can see that we have a very high matching score here.

Now let us talk about those values in the vector. For the above example, we simply used the frequencies of the terms as their weights, but there are better ways of weighing these dimensions. One popular way is TFIDF weighing. TFIDF is based on two simple intuitions:

1. The more the word occurs in a given document, the more important it is.
2. The less the word is common in the collection, the more important it is.

The first intuition is translated in the following formulation.

$$TF(t_i, d_j) = \frac{freq(t_i, d_j)}{\sum_k freq(t_i, d_j)}$$

Here, freq(ti,dj) is the frequency of term ti in document dj, and the denominator calcu- lates the size of that document. Thus, the frequency of a term is normalized by the size of a document.

The second intuition is translated can be translated as - the importance of a term is in- versely proportional to its frequency in the collection. This formulation of Inverse Document Frequency (IDF) is given below.

$$IDF(t_i) = log\frac{|D|}{|d_j : t_i \in d_j|}$$

Here, |D| is the total number of documents and |dj : Ti ∈ dj| is the number of documents in which term ti occurs.

Finally, we can combine these two calculations in to one score by multiplying them as shown below.

$$TFIDF(t_i, d_j) = TF(t_i, d_j) IDF(t_i)$$

Probabilistic Models

The probabilistic models estimate the probability that the user will find document d_j relevant to a user query q based on the assumption that such probability depends on q and d_j only. It also assumes that the user prefers a certain subset of all documents, which is the ideal answer labeled as set R and retrieved by maximizing the overall probability of relevance. Then the documents in R are predicted to be relevant to the query, while the documents not in the set are predicted to be non-relevant.

$$sim(d_j, q) \sim \sum_{i=1}^{t} w_{i,q} \times w_{i,j} \times \left(\log \frac{P(k_i|R)}{1 - P(k_i|R)} + \log \frac{1 - P(k_i|\overline{R})}{P(k_i|\overline{R})} \right)$$

* w_i,j ∈ {0, 1}, w_i,q ∈ {0, 1}

* R=set of initial guessed relevant documents, and $\overline{R}$ is the complement of R.

Similarly, the document list is sorted by descending order of degree of similarity for each document. To compute the probability of a document being relevant to the query, we can make simple assumptions like

$$\begin{aligned} P(k_i|R) &= 0.5 \\ P(k_i|\overline{R}) &= \frac{n_i}{N} \end{aligned}$$

, where n_i is the number of documents that contains the index term k_i and N is the total number of documents, to retrieve an initial set of relevant documents labeled as V. Then we can improve it by selecting a subset V_i, composed of documents that contain the term k_i, and write

$$\begin{aligned} P(k_i|R) &= \frac{V_i}{V} \\ P(k_i|\overline{R}) &= \frac{n_i - V_i}{N - V} \end{aligned}$$

. By recursively doing this process, we can improve our guess of the probability that a document is relevant to a query.

Language Models

Language models generate a probability distribution over a sequence of words that could be used to distinguish the difference between words and phrases. Estimating the likelihood of different phrases is widely used in the field of Natural Language Processing and has many applications such as machine translation, spelling correct, and speech recognition. It can also be used to predict the probability that a given phrase is in a given language.

In Information Retrieval, language models are used in the query likelihood model, which sees each document d as a “language” to compute the probability of a given query q is in the document’s language model M_d. Currently, the unigram multinomial language model is the most popular and successful model for M_d, which takes each query as a sequence of independent words. Then the query likelihood would be

$$P(Q|M_D) = \prod_{w} P(w|M_D)^{q_w}$$

where q_w is the count of word w in query q.

Another popular model is the multiple Bernoulli model which treats each query as a binary vector over the phrase. Then the query likelihood would be

$$P(Q|M_D) = \prod_{w \in Q} P(w|M_D) \prod_{w \notin Q} (1 - P(w|M_D))$$

where the first product is for words in the query and the second product is for words not in the query.