Vector Space Retrieval Models

The basic idea of the Vector Space (VS) model is to represent both a document and a query as a vector in a high-dimensional space where each dimension corresponds to a term. The main assumption is that if document vector V1 is closer to the query vector than another document vector V2, then the document represented by V1 is more relevant than the one represented by V2. That is, relevance is modeled through similarity between a document and a query. There are three fundamental problems in this approach: (1) how to define the dimensions (i.e., term space)? (2) how to map a document/query into a vector? (3) how to measure the similarity? Different answers lead to different variants of the vector space model.

Over decades, people have (heuristically) explored many variations of answers to all these questions. Empirical results seem to show that (1) single words are often good enough for defining dimensions, which is why single word-based indexing remains the dominant trend in retrieval and is what the current search engines are using; (2) TF-IDF weighting and document length normalization are among the most effective heuristics for computing term vectors; (3) the optimality of similarity measure highly depends on term weighting, which is not surprising as they are interacting components in the same retrieval function. Traditionally (before TREC), the most effective formula was believed to be TF-IDF plus cosine distance measure where TF doesn't involve document length normalization. In 1990's, largely due to the use of much larger data sets for evaluation in TREC, document length normalization was found to be quite important, which leads to the current version of TF-IDF weighting represented by the pivoted normalization formula and the Okapi formula. In both cases, dot-product has been found to be more appropriate than cosine measure.

TF-IDF weighting and document length normalization are quite easy to understand intuitively. However, how to implement them exactly in a formula is quite challenging and is still an open question. Empirically, people have found that some kind of sublinear transformation of the term frequency in a document is needed and incorporating document length normalization through the form "(1-s + s*doclength/avgdoclen)" (i.e., pivoted length normalization) is effective. This kind of normalization was justified in the a paper by Amit Singhal and others. Click here to read this paper. While BM25/Okapi and pivoted normalization are among the most effective ways to implement TF-IDF, it remains the single most challenging research question in information retrieval what is the optimal way of implementing TF-IDF. Read Fang et al. 04 for a recent discussion of this issue in the axiomatic retrieval framework.

An essential component of any retrieval model is the feedback mechanism. That is, when the user is willing to judge documents and label some as relevant others as non-relevant, the system should be able to learn from such examples to improve search accuracy. This is called relevance feedback. User studies have shown, however, a user is often unwilling to make such judgments, raising concerns about the practical value of relevance feedback. Pseudo feedback (also called blind/automatic feedback) simply assumes some top-ranked documents to be relevant, thus doesn't require a user to label documents. Pseudo feedback has also been shown to be effective on average, though it may hurt performance for some queries. Intuitively, pseudo feedback approach relies on term co-occurrences in the top-ranked documents to mine for related terms to the query terms. These new terms can be used to expand a query and increase recall. Pseudo feedback may also improve precision through supplementing the original query terms with new related terms and assigning more accurate weights to query terms.

In the VS model, feedback is often achieved using the Rocchio feedback method. In general, all feedback approaches try to modify the query representation based on feedback examples to obtain a presumably improved version of the query. The basic idea of Rocchio is simply to construct the new query vector (which is how you represent a query in the VS model) by moving the original query vector closer to the centroid vector of the positive/relevant document vectors and farther away from

the negative centroid. The actual formula has three parameters alpha, beta and gamma (see the slides), corresponding to the weight on the original query vector, the positive centroid and the negative centroid. These parameters need to be set empirically. In practice, negative examples are often not very useful, so in some versions, Rocchio may involve just moving the query vector closer to the positive centroid. For the sake of efficiency, the new query vector is often truncated to contain only k terms with the highest weights. In order to avoid overfitting to the relevant examples (often a small sample), we generally need to put a relatively high weight on the original query. The relative weight of the original query vs. information extracted from feedback examples often affects feedback performance significantly. In the case of pseudo feedback, setting an optimal weight is even harder as there are no training examples to tune the weights. How to do robust and effective pseudo feedback (related to how to optimize weighting of original query terms and new terms) is another open research question in information retrieval research.

BM25/Okapi plus Rocchio (for pseudo feedback) is generally regarded as representing the state of the art performance of retrieval.