Ramkumar Rajabaskaran 85241493: - <u>Review of Paper 2: -</u>

The paper talks about various successful techniques that were used match and index relative terms, to weigh the terms for an improved retrieval of more relevant data. These techniques were effective on simple terms for full texts and for large files of texts containing tons of documents.

The user's request is usually phrases, sentences or a word list from which relevant details are formed through the series of processes such as "Forming terms and matching", "Weighting" and "Iterative searching". The terms are usually the stems(roots) of the words and got after removing the stop words from a pre-determined list. This is done to make the process of creating terms more economical. Stemming using a standard stemming algorithm is done to remove the singular or plural forms and is reduces the term list. This improves the performance. Hence terms are formed this way.

Once the terms are formed, they are weighed by using a sum of weights from different methods such as Collection frequency, Term frequency and Document length. The idea behind weighting is selectivity, to select the relevant terms more frequently. Collection frequency places more importance to terms occurring sparsely than the more frequent ones. For a term **t(i): - CFW (i) = log N - log n**; where n=no. of documents t(i) occurs in and N=total no. of documents. **Term frequency** is the second case where it varies from one document to another. **TF (i, j) =** the number of occurrences of term t(i) in document d(j). The third input is the length of the document. It is defined as **DL (j) =** the total of term occurrences in document d(j). This is normalized by the average DL for all documents. **NDL(j)=DL(j) / Avg(DL).** This term that occurs the same in documents of different sizes is more valuable in this case.

Once we have the three kinds of weights, they are then combined. The combined weight formula is given by: - **CW (i, j) = [CFW (i) * TF (i, j) * (K1+1)] / [K1 * ((1-b) + (b * (NDL (j)))) + TF (i, j)], where K1, b are constants.** The values of K1 is chosen as 2 in this case after lots of tests and b=0.75. Documents are usually ranked on the decreasing order of their score, where the score of a document d(j) is the sum of weights of the query terms present in the document.

Iterative searching is a process that further improves the relevancy of a search result, by performing an initial search to obtain documents as per their search terms. The information obtained is then modified based on whether it is relevant or not. There are two methods in which this was done, Relative weighting and query expansion. In Relevance weighting, the search term weights are changed per its relevancy through the formula: - **RW (i) = log [((r+0.5) (N-n-R+r+0.5)) / ((n-r+0.5) (R-r+0.5))] where** r = the number of known relevant documents term t(i) occurs in, R = the number of known relevant document for a request. This can be used instead of CFW (i, j) for all terms in a second or subsequent search. The second method of query expansion ranks all terms from relevant documents per their Offer Weight. **OW (i) = r * RW (i).** Then the top 10 or 20 ranked terms are included in the search. Hence proper selection of more relevant terms is performed. The relevance weight may further be substituted in the Iterative combination to get a better formula **CIW (i, j) = [RW (i) * TF (i, j) * (K1+1)] / [K1 * ((1-b) + (b * (NDL (j)))) + TF (i, j)].** Longer queries however have an adjusted formula to get the adjusted combined weight **QACW (i) = QF(i) * CW (i, j) or** Query Adjusted Combined Iterative Weight**: QACIW (i) = QF(i) * CIW (i, j).**

These are the methods to tokenize and weigh words to obtain relevant searches.

The paper talks about the various methods to form tokens, by removing stop words and to stem words to obtain an economical list of tokens. It then takes us through the various ways to weight the tokens and how to improve the token's relevancy through the concept of relevance weighting and query expansion. This is however for only simple tokens and is not recommended for complex lexicons used as search terms.