**Implementing Query Expansion Techniques**

Our team decided to implement query expansion techniques in the form of thesaurus expansion and Rocchio algorithm.

**Thesaurus Expansion**

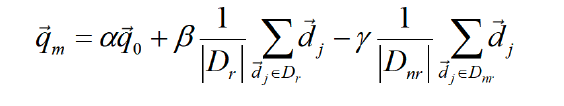
Thesaurus expansion in our project is done via the WordNet corpus from NLTK. Using that, we can easily find synonyms for every word in the query. However, it must also be acknowledged that using query expansion for every term in the query can be a bit excessive, especially if said query is very long. This is one of the drawbacks of query expansion that we learnt in lecture.

To mitigate this, and to search for the words which seem more likely, we try to find the query weight, via one additional round of cosine scoring on the term. Although this may seem like a lot of work, we get a good idea of which terms we should be expanding. Then, we can only choose to expand the terms which meet a certain threshold, as these are the terms which are judged to be more important. This way, we avoid the issue of expanding irrelevant or terms which are perhaps less related to the query at hand.

Admittedly, there are some drawbacks to this approach, namely that there is an assumption that the system works in the first place, else the terms which we have expanded may not be the relevant ones.

**Rocchio (Relevance feedback)**

The original Rocchio algorithm takes the form of:



For the purposes of the project, we ignore the irrelevant documents centroid, since it is going to be comprised of the majority of the documents that we have.

Instead, we only focus on the relevant vectors. Normally, for these document vectors, the vector used (Dj) is generated using the document count vector. However, in the interest of saving space, we only store the top 20 terms (in terms of frequency) of each vector. We do this using the assumption that anything after that it is unlikely to appear too many times in the document and hence would not affect the score much. Of course, this is an approximation and definitely does not hold true all the time, but we felt that the trade-off was acceptable. The query vector was computed as per the lecture.

A rough algorithm of what we did is given below:

1. For each term of the query, calculate the centroid of the new term in the new query vector. You can think of this as vector summation along a single dimension (row) of the vector.
2. There are some terms that can be found in the relevant documents but not in the query. To account for this, we get the set of terms which were not seen in the query, and then calculate the centroids for those terms as well. We then do the normal cosine scoring (via dot product) on these terms with the postings.
3. Normalization is done before hand, as we have saved the magnitude of each document weight vector during the indexing step.

The parameters for the Rocchio algorithm were varied according to our experimental data and what we thought fit the leaderboard the best.