We experimented with 3 types of query expansion techniques:

### Relaxing boolean and phrasal queries

The first stage of query expansion, as explained above, involves relaxing the restrictions placed on the query from phrases and boolean operators. Since the terms in the user queries may not be the exact terms desired, we need to relax the AND portion of the query, so that even if the term given is not correct, the results for other parts of the query can still be returned. To achieve a baseline tf-idf framework, all boolean operators and phrase markers were stripped from the query string.

Every query can one of the following four types:

1. +phrase, + boolean: e.g. "fertility treatment" AND damages

2. +phrase, -boolean: e.g. "fertility treatment" damages

3. -phrase, +boolean: e.g. fertility AND treatment AND damages

4. -phrase, -boolean: e.g. fertility treatment damages (basic free text query without phrases or boolean operator)

For an original query string with both phrases and the "AND" boolean operator, query expansion can allow us to relax these restrictions in order to produce the other 3 combinations. When the original query either does not have phrases or the boolean operator, it can still be relaxed to the free text query. For a maximally complex query of type 1 (including boolean operator and phrases), the orders of search between these four types of queries can be permuted and experimented with to determine the importance of preserving the additional information of phrases and boolean operators.

### WordNet/Thesaurus Query Expansion

The NLTK WordNet was used as a thesaurus to implement query expansion. In particular, the synonyms of the terms were found using the synset feature of WordNet. Additionally, we also experimented with using hypernyms and hyponyms to return related words, however, because too many of such words were returned, we decided to stick with synonyms. The additional synonyms retrieved were appended onto the free text version of the original query to create a longer free text query. Due to time constraints we were unable to implement a potential improvement, which involves ensuring for a boolean query that the synonyms were intersected. For example, a query 'quiet AND phone call' could be expanded to '(quiet OR silent) AND (phone call OR telephone call)'.

### Relevance Feedback and Rocchio Algorithm

For relevance feedback based on the Rocchio Algorithm, our system makes use of the top 1000 returned documents from the basic search which are assumed to be relevant, on top of the list of documents identified as relevant in the original query file. The document IDs are then used to retrieve precomputed and stored document vectors in the document properties file, which are then combined to give the centroid vector of the relevant documents. This is done such that there is no need to traverse the postings file to build the document vector for each relevant document, which would be extremely expensive.

An additional optimisation involves storing the vectors as sparse vectors using a dictionary mapping terms to the tf values. This is necessary since the vectors would include many 0 terms if the dimension of the vector was the size of the entire dictionary. Furthermore, even after computing the centroid vector, there will still remain many non-zero dimensions in the centroid. In order to improve on efficiency, there is a need to remove some of the non-zero terms. To do this, each component was multiplied with idf in order to reduce the value of more common and hence less useful terms. The top 50 terms were chosen for the final centroid vector.

The new centroid vector can then be added to the original query vector to derive a new query vector used for VSM retrieval. For simplicity, the original query vector is made to be a free text query such that boolean operators are removed and phrases are converted to single word terms. The additional documents found from relevance feedback are appended after the already returned documents.

## Experimental Results

F2 results for documents appended in the following order:

1. positive list 2. -boolean, -phrase 3. +boolean +phrase 4. +phrase -boolean

Q1 Average F2: 0.0108201093105916

Q2 Average F2: 0.362745098039216

Q3 Average F2: 0.0112925624835047

Q4 Average F2: 0.496296296296296

Q5 Average F2: 0.103200491131526

Q6 Average F2: 0.30050505050505

Mean Average F2: 0.214143268

This is close to the baseline tf-idf.

F2 results for documents appended in the following order:

1. positive list 2. +boolean, +phrase 3. -boolean, -phrase 4. +phrase -boolean

5. Wordnet expansion 6. Rocchio expansion

Q1 Average F2: 0.0318471337579618

Q2 Average F2: 0.276839007986549

Q3 Average F2: 0.00854730742939909

Q4 Average F2: 0.516624579124579

Q5 Average F2: 0.103200491131526

Q6 Average F2: 0.172659817351598

Mean Average F2: 0.184953056130269

This performed worse than the baseline tf-idf.