# Bonus.docx

## Indexing

Documents were divided into 4 parts, title, content, date and court and do trigrams for all. But this resulted in a crazy amount of unique words in the dictionary. As a reference, implementing bigrams from unigrams resulted in 5.5million terms which is 5million more than the 500 thousand I get from purely unigrams. Nonetheless this approach was experimented with to improve the search-time performance for Boolean queries

## Boolean Search

The area of lnc.ltc scoring for Boolean search was an experimental one that is aimed to yield a better and more representative result. On the query side, ltc was used. But since each constituent query term was treated as a mono/bi/trigram, it resulted in ltc being a constant multiplier for each constituent. During AND-merging, scores of matching documents were multiplied by one another instead of summing. This retained the effect of normalising (0 < score < 1) to give a more representative final result.

## Free-Text Search

A tiered approach was attempted in the context of a law article retrieval. During indexing, term frequencies were distinguished between court name, title, date and content. These respective term frequencies were handled independently for the scoring scheme. The tiered scoring scheme awarded a point system depending on the term frequencies in each tier. For example, content and date had a base score of 1. Matching docID, court, or title would give a base score of 2. The resultant list is then assessed in the order [docID, court, title, date, content] – of descending importance. The lists are then sorted by their natural order to reflect the relevance of the document to the query. Hopefully this method allows for a more “human” or natural form of assessing relevance to the search query.

## Query Expansion

We try to use nltk’s corpus WordNet to generate the query’s synonyms. But we find it is not working as expected. First, it can not work on Boolean Search. There is two way we treat the new word we introduce by Princeton’s WordNet. Treated as OR or Treated as AND. If we treat every synonym we introduce as a AND term, then the return documents will decrease sharply. If we treat every synonym we introduce as a or term, because sometimes, some common words are very likely to be generated as a synonym, so it will retrieve many additional irrelevant documents. For Free-Text Search, it will cause the same problems as AND case. For example, ‘Apple Inc.’ may generate ‘Malus pumila Inc.’ which we don’t want to see. In conclusion, we don’t involve Query Expansion, but we try and find it’s not very useful for this project.