Query Refinement Techniques Explored

In this project, we explored four query expansion techniques, and combinations of them:

1. Query expansion using NLTK’s Princeton WordNet
2. Pseudo-relevance feedback using Rocchio [Okapi System]
3. Spelling Correction through distance checking
4. LESK Algorithm for context related expansion

This document briefly outlines the query expansion techniques implemented, the algorithm and libraries used and how these techniques affected our performance.

The overall idea was to expand our query so as to search within more possible documents. The query term generated using different techniques were given different weights to be incorporated based on importance that was later used in ranking.

Due to less test queries and not having many opportunities to run on leaderboard framework our MAF scores to understand the effect of these techniques might be due to other regressions and changes.

**NLTK’s WordNet**

NLTK corpus’s WordNet is an English lexical database. It is used to expand all terms in free text queries in the following way:

1. We used Princeton’s wordnet to generate synonyms
2. With free text queries we found synonyms for all words
3. With Boolean queries we converted it to free text queries and used evaluation of free text queries.

## Analysis:

* On first trying to find results for all synonyms we were getting a lot of results leading to low precision, hence we decided to limit the number of synonyms for each term and tuned a parameter by making it 40%.
* We also had to process certain terms as they contained characters like underscore and later found that some synonyms were actually duplicates and hence had to incorporate that as well.
* Most terms we found weren’t really the right substitutes for the query term and hence decided to place lower weight for documents which were ranked using these terms.
* This also led us to the expansion technique of context relation expansion using LESK algorithm

Overall, this helped us in finding correct documents but returned a lot of results. By tuning our parameters and their weights we could effectively use this technique to get a good MAF.

**Context Related Expansion: LESK**

After some improvements in score with Wordnet, we wanted to experiment with generating terms which are related to context of query and give them a higher weight compared to WordNet synonyms.

The following algorithm was used:

1. Use the lesk function from nltk.wsd
2. For each query term generate synonyms with the original query as context.
3. Add these terms to query with higher weight compared to words from Wordnet

## Analysis

* This technique was very useful as it helped rank the documents much better and improve our scores drastically (along with other improvements)
* Compared to using wordnet, this was much powerful as it generated context relevant synonyms and very few of them which it deemed important.
* This technique is much better than using all synonyms for each query term as it leads to query drifting giving unexpected results

A screenshot of a social media post

Description automatically generated

Average MAF vs % of synonyms taken into query expansion

Overall, query expansion with LESK performed better but it wasn’t working well if used alone hence we still used WordNet and gave higher weights to query terms generated from LESK as compared to WordNet. This led to all correct results being returned and LESK helped in ranking them in an effective manner.

We decided to keep this as this helped boost the MAF2 score to 0.196.

**Pseudo-Relevance Feedback**

The following algorithm was used to expand our query based on Pseudo Relevance feedback:

1. The first step was to process our results based on the tf-idf ranking for free text and by using extended Boolean model for phrasal queries.
2. Once we got the results, we used the top 3 documents as relevant documents by assuming that these are the most relevant.
3. The top 10 terms were then taken from these documents based on document frequency. [Inspired by the Okapi System]
4. These terms were then added back into the query terms and were given initial weights based on Rocchio formula before tf-idf ranking was done on them [ Similar to how weights were assigned for new terms found through query expansion]

## Analysis of PRF:

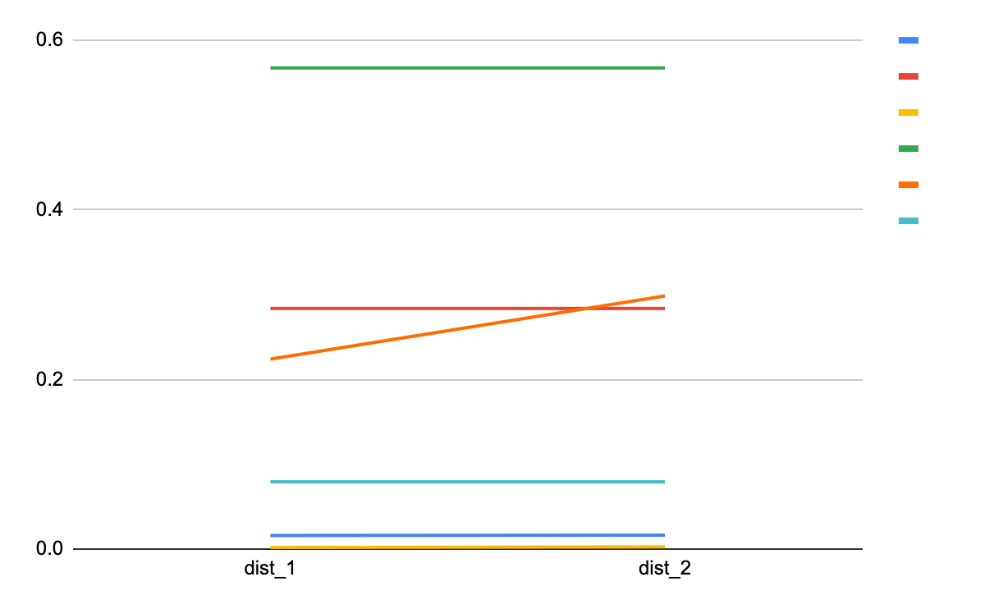
* After a few experiments with test queries to find the best APLHA, BETA and GAMMA for Rocchio’s formula and then testing on leaderboard with that we decided not to use this. The main reason was that it gave a lot of results and reduced our precision. Moreover, the documents termed as “relevant” in pseudo manner were mostly not as expected and hence this skewed the results to rank correct documents even further down.
* Not having enough test queries and attempts at running using leaderboard, we couldn’t tune the parameter and weight scores and hence weren’t confident in it giving us better results.
* We also realized that using PRF in legal corpus might not be of much use as the most common terms were present in a lot of documents. In future we would have wanted to test this by finding the relevant terms by using tf-idf and then ranking these to get the top 10 terms and hence avoiding stop words and common words.
* Since indexing and postings were stored in inverted manner, it was a time and memory consuming process to create document vectors for relevant documents. This also prevented us from taking a large number of top documents as relevant since it would be time consuming.
* We also chose to ignore non-relevant documents since we found this to not help much as we were eventually making pseudo assumptions which may give incorrect results.
* On attempting doing only PRF without any other Query expansion techniques we found it to be useful as it helped in finding correct documents with the expanded query especially for Boolean queries.
* With better searching technique in initial phase we feel that PRF can help in improving the search engine drastically.

On our first attempt of using Rocchio we got an MAF2 of 0.05 which fell drastically from 0.137 which made us realize it wasn’t helping. We gave it another try towards the end when our MAF was closer to Baseline and during that the MAF wasn’t impacted much and hence decided not to use it for submission as it’s time consuming and also not helping in boosting the score.

**Spelling Correction**

To account for spelling errors like “quite” vs “quiet”, we use Minimum Edit Distance for spelling correction using the Pyspellcheck library. We decided to use a maximum edit distance of 2, and add the top 3 suggestions that have the highest word probability to our expanded query. This restriction is to avoid adding unknown vocabulary terms and to prevent the context of the sentence from changing, in case there is no actual misspelling.

## Analysis:



We can see from the above diagram that for the 6 test queries the MAF either remained same or increased when spelling correction was done with distance of two instead of one. Hence spelling correction helped a lot.

**Conclusion**

Overall, we found that certain query expansion techniques helped us in not only improving recall by finding more documents but also helped in ranking them properly.

We also concluded that for phrasal Boolean queries and smaller free text queries it was very useful in generating better results as it searched for more documents for other possible terms the user could have used. Moreover, using a law thesaurus would have been helpful as we know that the users searching are lawyers in this case.