**Query Refinement (BONUS)**

|  |  |  |  |
| --- | --- | --- | --- |
| **A0189690L** | **A0190198W** | **A0199786X** | **A0200679A** |
| [e0325274@u.nus.edu](mailto:e0325274@u.nus.edu) | [e0325782@u.nus.edu](mailto:e0325782@u.nus.edu) | [e0406767@u.nus.edu](mailto:e0406767@u.nus.edu) | [e0407660@u.nus.edu](mailto:e0407660@u.nus.edu) |

We explored two main query refinement techniques - query expansion and relevance feedback. The results of applying these techniques on a reduced dataset used for testing are summarised in the table below.

|  |  |
| --- | --- |
| **Technique(s) Tested** | **Run Time Performance** |
| None | Completed in 0.5674 seconds |
| Query expansion | Completed in 1.7522 seconds |
| Relevance Feedback v1 | Completed in 66.1701 seconds |
| Query expansion + Relevance Feedback v1 | Completed in 66.4005 seconds |
| Query expansion + Relevance Feedback v2 | Completed in 1.8086 seconds |

We adopted a thesaurus-based query expansion method, using WordNet from the nltk module. Using the parsed query terms (i.e. separated into individual tokens or phrases), we generate the synonyms for each term from the thesaurus and append them to the respective segments in the array of query terms. We then use the updated parsed query terms (including synonyms) to calculate the cosine scores of the eligible documents.

As a result of query expansion, the run time of search has increased. Using a trimmed version of the given data set, the run time has increased from 0.05 seconds to 1.8 seconds (average approximation across multiple runs). We feel that this is a reasonable trade-off, for we are able to consider more (potentially relevant) documents that contain words that have the same meaning as the query terms, and not just strictly terms that appear in the query.

However, we are also aware of the limitation of query expansion. While recall will increase as more (potentially) relevant documents are retrieved, precision may decrease when the query terms are ambiguous or when the retrieved synonyms do not fit into the context of the query.

Perhaps, to further improve on the score weighting, we can give a higher weight to the actual query terms and a lower weight to the expanded query terms. This will allow us to give a higher score to the documents that contain the actual query terms.

Additionally, we also adopted a pseudo relevance feedback method (v1). We extracted important terms from the documents bookmarked as relevant and added them to the query. We determined these important terms by looking at their linguistic tags using NLTK’s pos\_tag method. Generally, we found adjectives, nouns and verbs important, ignoring terms tagged as prepositions, adverbs, and pronouns.

The tags below are what we passed to NLTK’s pos\_tag method. Each category (adjectives, nouns, verbs) have a few variants, such as plural forms, that are also covered by the tags.

REL\_TAGS = ['JJ', 'JJR', 'JJS', 'NN', 'NNS', 'NNP', 'NNPS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']

These important terms were then added as values under the document in a dictionary and used for the searching phase. However, testing highlighted that our naive method was highly inefficient, taking almost 60x longer. This was because there was a large number of “important words” being identified, an average of 1,500 per document.

We thus embarked on a v2 that curated that list of important words for each document to 5. This not only would result in a more accurate result (as it would be more likely that the refined words were more relevant to the query), but would reduce our search time greatly as well. To choose the 5 words that would be considered relevant for each document, we ranked the words in the document by their term frequency score. This is accomplished during indexing, so that during search the 5 relevant terms per document could be immediately retrieved without any additional calculations.