**Exploration on query refinement**

Code for query refinement is included in `query\_refinement.py`

We explored 2 query refinement techniques:

1. **Pseudo-relevance feedback using Rocchio algorithm**
   1. Only performed for free text queries, as we assume that if the user is searching for an exact match (using phrasal or boolean queries), they will not appreciate receiving results with documents which do not exactly match their query.
   2. If toggled, the preliminary result from the basic search (or after query expansion, depending on whether that is toggled on) will be re-evaluated.
   3. Firstly, all the documents are indexed with the top 100 (or less, depending on the maximum number of unique tokens in the document) and their corresponding weights. This forms the foundation for the vectorisation of each document.
      1. While only storing 100 tokens for each document might not be very accurate, it is a reasonable compromise between storage constraint and accuracy.
      2. These documents are referred to as `trimmed\_documents`.
   4. Next, when Rocchio is toggled on, the top 50 documents are assumed to be the most relevant. This number of documents is rather arbitrary, but we think it is a reasonable size of documents to use to find the centroid.
   5. The tokens and relevant weights are extracted for each of these documents, and the centroid is found by summing the weight of each token across all documents, then dividing of them by the number of documents. This is much like finding all the “midpoints” of the tokens (employing the same concept as coordinate geometry) and then putting the values of the dimensions together to form a new vector.
      1. This is the centroid.
   6. The original query is also vectorised by filling each dimension of token with their corresponding weights.
   7. The original vector is then modified with the centroid, based on the Rocchio formula, using weights recommended by Stanford NLP
   8. The resulting new query vector is then used to run cosine-simlarity scoring with all the trimmed documents .
   9. The resulting list is sorted by cosine score in descending order, and the corresponding list of docIDs are extracted and returned as an updated ranking of the relevant docIDs.
2. **Query expansion using synonyms from Wordnet**
   1. Only performed for free text queries, as we assume that if the user is searching for an exact match (using phrasal or boolean queries), they will not appreciate receiving results with documents which do not exactly match their query.
   2. If toggled, the query is expanded before being evaluated during the first round of retrieval.
   3. Only the synonyms of the nouns in the free text queries are looked up on Wordnet. Out of these synonyms, only synonyms which are nouns are added to the original query.
      1. The parts of speech for each token in the query are tagged using nltk’s pos\_tag method. Only nouns (those tokens whose pos tags begin with ‘NN’) from the query are filtered.
      2. The synonyms for these nouns are retrieved using nltk’s synsets method. Only those synonyms that are nouns are added to the original query.
   4. Why choose only nouns and disregard other parts of speech e.g. adjectives or verbs?
      1. Initially, we expanded every term in the query. However, this added irrelevant terms to the query.
      2. For example, for the query ‘quiet phone call’, synonyms of the adjective ‘quiet’ such as ‘lull’, ‘tranquility’, ‘silent’ and for the query ‘good grades exchange scandal’, synonyms of the adjective ‘good’ such as ‘effective’, ‘dependable’ were added to the query. We judged these synonyms to be irrelevant, and after trying a few more queries with a mix of adjectives, nouns, verbs, and adverbs, determined that only the synonyms for nouns seemed to be relevant.
      3. For example, for the query ‘quiet phone call’, ‘telephone’ was added as a synonym of ‘phone’. After adding ‘telephone’ to our query, the ranking of the document with id ‘6807771’, which was assessed as relevant by Intellex, increased. Earlier, this document was ranked very low in our results. But after adding ‘telephone’, its ranking improved such that the document was ranked within the first half of the total number of retrieved results.
   5. One limitation of query expansion is that irrelevant nouns of certain ambiguous words get added to the query and skew the results.
      1. For example, for the query ‘good grades exchange scandal’, running the query without query expansion yielded both documents assessed as relevant by Intellex in the top few results. When the query was expanded, words such as ‘central’ (for ‘exchange’) and ‘rally’ (for ‘scandal’) were added. This resulted in a case on an election rally being ranked the highest, which is irrelevant to the original query.
      2. Because of this limitation, we decided to toggle query expansion off.