Algorithm Components

There are five major components of the vector space retrieval model:

1. Preprocessing
    a. *Tokenization*: called nltk word_tokenize( ) function to tokenize terms from each document in the collection
    b. *Stop words and punctuation removal*: the corpus of stop words is consisting of stop words text file provided in the Time file folder, numbers, common punctuations and symbols and nltk's collection of stopwords. The finial stop words list is the union of three sources. The removal step also performed on each document in the collection.
    c. Lemmatization and stemming: each token in the documents from the above step undergo the morphological analysis to return the base form of a word
    d. Normalized term frequency: each document becomes a dictionary of a token and normalized term frequency pair. To account for difference document lengths, normalization on the raw term frequency was calculated using the formula: raw term frequency/ length of the document

2. TFIDF weighting of terms (Jupyter NB TFIDF functions section)
    a. *Document frequency (DF)*: each word from all the documents was counted on the number of documents it has appeared in.
    b. *Inverse document frequency (IDF)*: calculated by the formula—$\log_2(423/DF)$, 423 being the total number of documents in the collection
    c. TFIDF weights: normalized term frequency*IDF will overweigh some rare terms and under weigh the more frequent term

3. Query processing
    Similar to preprocess documents, each query term is tokenized, stemmed and calculated a normalized term frequency. As the result of observation that most of the query term appear once in the query, query terms scores are not weighted by tfidf.

4. Vector space retrieval model
    a. *Retrieve relevant documents*: when a query is issued, the function will find the documents that contain at least one of the query terms
    b. *Document-term vector*: for each retrieved document, the function then extracts the term and its corresponding TFIDF score and put them into a vector then followed by filling in zeros to match the query vector length
    c. Cosine similarity: calculate cosine similarity between the query vector and each document-term vector. The result of such serves as a score for ranking the

document relevance. Because the term frequency was normalized prior, score normalization was not performed. The TFIDF values are in decimal places leading to even smaller cosine similarity values, thus, the final score was multiplied by 10^18 for easier comparison by human

d. Ranking: order the score descendingly to rank the retrieved documents

5. Input and output
   Reading documents in real time:
   a. Input: ask user to type an integer which is the index of a query
   b. Function: go through all the above steps
   c. Output: top 10 documents that have the highest similarity score with the query

Discussion

Distinguish documents that have the same similarity score

The algorithm of the project cannot evaluate documents that are score the same. The Word2Vec or a probability model can be inserted in the algorithm in the future to help discriminating document relevance. Relevancy and recall tests were not performed due to in the benchmark relevance file, some results do not include any of the query terms. Besides, the numbers of retrieved documents for the queries from benchmark and algorithm output are different. The algorithm needs to improve on thresholding a relevance score in order for a document to be considered relevant. Rather than having 10 results for all queries.