Assignment 2

-Shruti Kumari
-2018CS50420

Dataset parsing

A vocab was generated for all the documents which were in the top100 ranked documents for any of the query which was used to make tf vectors. The idf for each word was generated once in the beginning. Since generation of vocab and idf vectors take time, I have generated them and stored them in a directory named 'stored' in the submission. If need be, one can use these files to test queries rather than generating the idf again and again.
The content for each cord uid is parsed using the abstract, title and one of the pmc_json, and pdf_json files (pmc_json is preferred if available). To generate the terms, multiple methods were tried:

- Tokenizing the content using nltk.word_tokenize: This was not preferred because it does not take care of delimiters and considers symbols such as ';' terms.
- Tokenizing using re.split: To tackle this, we split the content using the following delimiters: `r'[\n`;,\s.:"\'\[\]\(){}]\s*'`
    - Stemming using Porter stemmer
    - Stemming using Lancaster stemmer

  I expected that Lancaster stemmer would give better results, but with this dataset, Porter stemmer was doing better reranking, hence Porter stemmer is used finally.

Vocab for relevant documents and vocab for all documents for these three ways to generate terms can be found in the 'stored' directory. Could not upload this 'stored' directory to moodle as the size was too large.

Rocchio Reranking

The top 100 documents were re-ranked using the Rocchio method. For each query, an optimised query was formulated using the term frequency vectors of all the relevant documents.
After the query is optimised, its similarity with each document is determined using cosine similarity and the documents were ranked according to this score. Cosine similarity was calculated between the tf-idf vectors of the optimised query and each relevant document.

(alpha,beta)	**ndcg@5**	**ndcg@10**	**ndcg@15**	**mrr**	**map(2)**	**map(1 & 2)**
Original:	0.50	0.48	0.49	0.53	0.35	0.51
(1,0.75)	0.49	0.48	0.47	0.58	0.34	0.51
(1,1)	0.49	0.48	0.47	0.58	0.34	0.51
(0.66,1)	0.49	0.49	0.46	0.58	0.34	0.51
(0.5,1)	0.49	0.49	0.46	0.58	0.34	0.51

(0.25,1)	0.48	0.49	0.46	0.58	0.34	0.50
(1,0)	**0.58**	**0.57**	**0.56**	**0.75**	**0.38**	**0.54**

It is visible from the above results that my implementation is comparable to the rankings we were provided with. Since the model is performing best for (alpha,beta) = (1,0.75) when we optimise the query, we set the hyperparameter alpha as 1 and beta as 0.75.

Observation: An unexpected observation was noted that when the query is not optimised, it is giving much better results than even the original rankings we were provided with.

Language Modelling

Using models suggested by Lavrenko and Croft, the top 100 documents are reranked. In this we also needed to maintain the frequency of terms in the whole collection, unlike just the document frequencies of terms in Rocchio. P(w,q1..qk) is the only thing which is calculated differently in iid sampling and conditional sampling. Since in iid sampling, we are making a very strong assumption we get far better results with RM2 than RM1. Smoothening is done using Dirichlet smoothing. The results obtained by both the methods are tabulated below for various values of mu.

RM1 (mu)	**ndcg@5**	**ndcg@10**	**ndcg@15**	**mrr**	**map(2)**	**map(1 & 2)**
Original:	0.50	0.48	0.49	0.53	0.35	0.51
1	0.52	0.51	0.53	0.65	0.36	0.50
1.5	0.52	0.53	0.53	0.65	0.36	0.50
2	0.53	0.53	0.53	0.67	0.36	0.51
2.5	0.53	0.54	0.50	0.64	0.37	0.51
0.5	0.52	0.53	0.50	0.67	0.36	0.51

RM2 (mu)	**ndcg@5**	**ndcg@10**	**ndcg@15**	**mrr**	**map(2)**	**map(1 & 2)**
Original:	0.50	0.48	0.49	0.53	0.35	0.51
1	0.53	0.50	0.52	0.55	0.36	0.52
1.5	0.54	0.51	0.52	0.56	0.36	0.52
2	0.54	0.53	0.52	0.61	0.37	0.53
2.5	0.54	0.54	0.51	0.58	0.37	0.53

0.5	0.53	0.49	0.50	0.52	0.35	0.51

It is clearly visible from the above results that both RM1 and RM2 are performing better than the original ranking we were provided with. Since the model is performing best for mu = 2, we set the hyperparameter mu as 2.