COL764 Assignment-2

Navya Jain

2019CH10106

1 Introduction

The aim of the assignment is to develop “telescoping” models aimed at improving the precision of results using pseudo-relevance feedback.

1.1 Dataset

1. **Document Collection**: There are $192,510$ unique CORD identifiers in the collection, with each CORD id corresponding to one or more full-text files.
2. **Queries**: 40 topics with three fields from the TREC COVID19 track. In the scope of this assignment used field **query** for Language Modeling and field **question** for Rocchio’s method.

2 Program Structure

2.1 Pre-Processing Dataset

For the calculation of TF-IDF, a complete pass on the dataset is required. To improve the retrieval quality the following document pre-processing scheme was adopted during the pass:-

1. **Tokenization** : Regex based pattern matching is used to split the text fragments at the following symbols :- $[', .- : (); ?“”]$ to obtain the tokens.
2. **Stopwords** : Stopwords are the most common words of the collection which do not add much information to the collection. These are filtered out from all the tokens obtained post tokenization. Set of stopwords to be removed are obtained from the *nltkpopular* library.

3. **Length of token** : A lot of single characters were found in the document collection. These might be representing symbols in the research paper but do not add much importance in our setting. Also, double lettered words such as *et* and *al* are found in almost all the research papers. These words as well do not add much importance for us. Therefore, we set a filter all the tokens with length greater than 2.

4. **Numbers** : Numbers are removed from the collection. Although numbers might be relevant for queries seeking years and dates, there seems to be a trade off between runtimes and retrieval quality.

2.2 Part 1 : Rocchio's method

Rocchio's Method aims at finding a query vector that maximizes similarity with relevant document and minimizes similarity with non-relevant documents. The Query reformulation moves the query vector closer to the centroid of the relevant documents and distant from the centroid of the non-relevant documents.

(insert photo of centroid dots etc)

The following terms are required for calculating the vector representation of a document :-

1. **Document Frequency DF** :- DF is required to calculate the weight of a term in the vector representation of a document. The number of documents in the entire collection containing a term i is referred as df_i. This is stored in a **dictionary** with keys as the terms and the values as the number of documents containing the term. For each document a set is created for all the tokens present in the document, and the values of these tokens are updated in the dictionary. It takes about $8mins$ to prepare the document frequency dictionary of the entire collection. Another dictionary mapping the cordids to their pmc and pdf json files is also stored in memory.

2. **Term Frequency TF** :- TF of a term is required to calculate the weight of a term in the vector representation of a document. The number of occurences of a term i in a document d_j is referred as the tf_{ij}.

3. **Weight of Term** : The weight of a term i in document d_j is referred as w_{ij} and is calculated as the product of tf_{ij} and idf_i. where $tf_{ij} = 1 + \log_2 f_{i,j}$ and $idf_i = \log_2 \left(1 + \frac{N}{df_i}\right)$.

   $$w_{i,j} = \begin{cases} TF \times IDF, & \text{if } TF > 0 \\ 0, & \text{otherwise} \end{cases}$$

Vector Representation :- For all the terms of the vocabulary, the weights are calculated as described above and presented as $\overrightarrow{d_j} = (w_{1,j}, w_{2,j}, w_{3,j}, \ldots, w_{t,j})$ $\vec{q} = (w_{1,q}, w_{2,q}, w_{3,q}, \ldots, w_{t,q})$ for the vector representation of a document d_j and query q.

2.2.1 Query Reformulation

For pseudo-relevance feedback, we re formulate the query provided to us according to the Rocchio's formula as follows :-

$$\overrightarrow{q_m} = \alpha \overrightarrow{q_0} + \beta \frac{1}{|D_r|} \sum_{\overrightarrow{d_j} \in D_r} \overrightarrow{d_j} - \gamma \frac{1}{|D_n|} \sum_{\overrightarrow{d_k} \in D_n} \overrightarrow{d_k}$$

1. **Query Vector** : Query vector is calculated for the given query using the tf-idf product.

2. **Relevant Document Vector** : It is assumed that the top-100 documents provided to us are all relevant to the information need. A vector concatenating all the document representation of relevant documents is created.

3. **Non Relevant Document Vector** : A set of 1000 documents are randomly chosen from the entire collection, representative of the non-relevant documents present in the collection to the query. This is pre-computed and stored in memory for all the queries, since 100 documents won't budge the centroid to a great extent.

2.2.2 Re-Ranking

The cosine similarity of the formulated query is calculated with the top-100 documents and the list is reverse sorted according to the obtained similarity scores.

$$\text{sim}(d_j, q) = \cos(\theta) = \frac{\overrightarrow{d_j} \cdot \vec{q}}{|\vec{d_j}| \times |\vec{q}|}$$

2.2.3 Evaluation Metrics

The evaluation scores obtained for the reformulated query calculated using treceval is as follows :-

Roochio		
	Original Ranking	Re-Ranking
nDCG @ 5	0.4844	0.5665
nDCG @ 10	0.4833	0.5425
nDCG @ 30	0.4613	0.5018
nDCG@ 100	0.4042	0.4165
MAP	0.0576	0.0597

2.2.4 Hyperparameter Tuning

Two grid searches are performed over a range of α, β, γ values. Highest Average nDCG score was found at $\alpha = 0.8, \beta = 0.2, \gamma = 0.2$.

$$nDCG[k] = \frac{DCG[k]}{DCG'[k]} \text{ where } DCG[k] = \sum_{i=1}^{k} \frac{G[i]}{\log_2(i+1)}$$

1. **Grid Search 1** :-

a	b	y
0.5	0.45	0.15
0.6	0.55	0.15
0.7	0.65	0.15
0.8	0.75	0.15
0.9	0.85	0.15
	0.95	0.15
	1.05	0.15

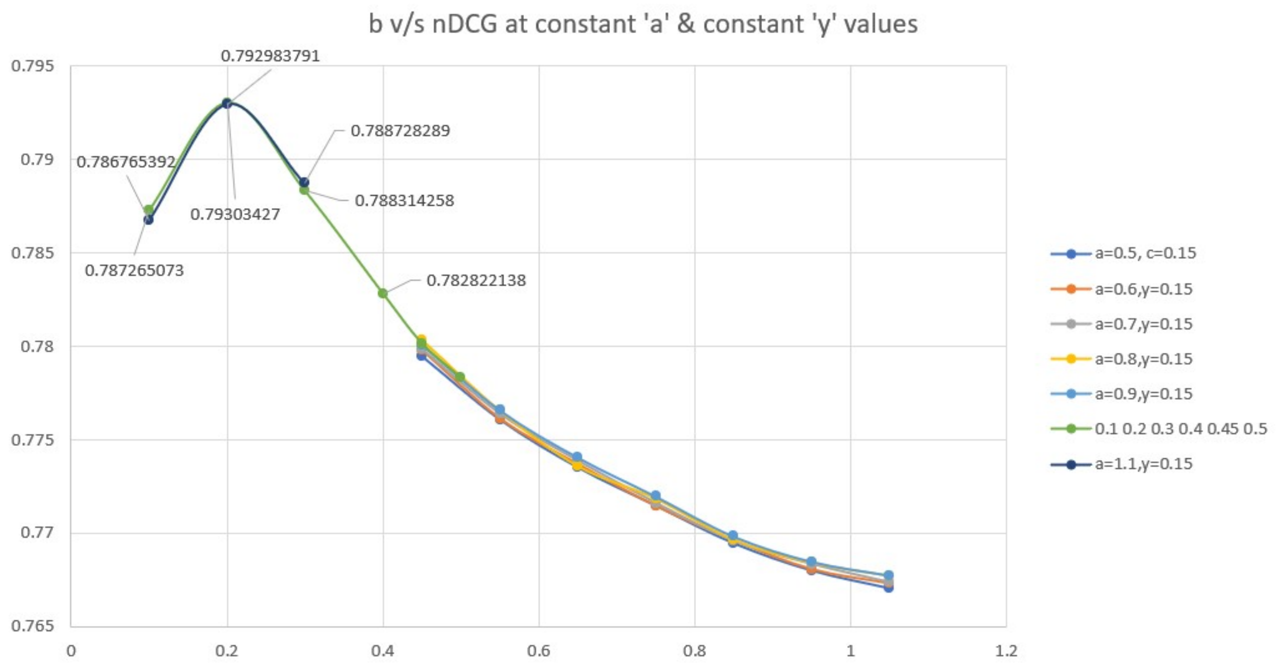

Figure 1: b v/s nDCG score at constant 'a' and constant 'b'

2. **Grid Search 2** :-

Grid Search -2		
a	b	y
0.7	0.1	0.05
0.8	0.2	0.08
0.9	0.3	0.11
1	0.4	0.14
1.1	0.5	0.17

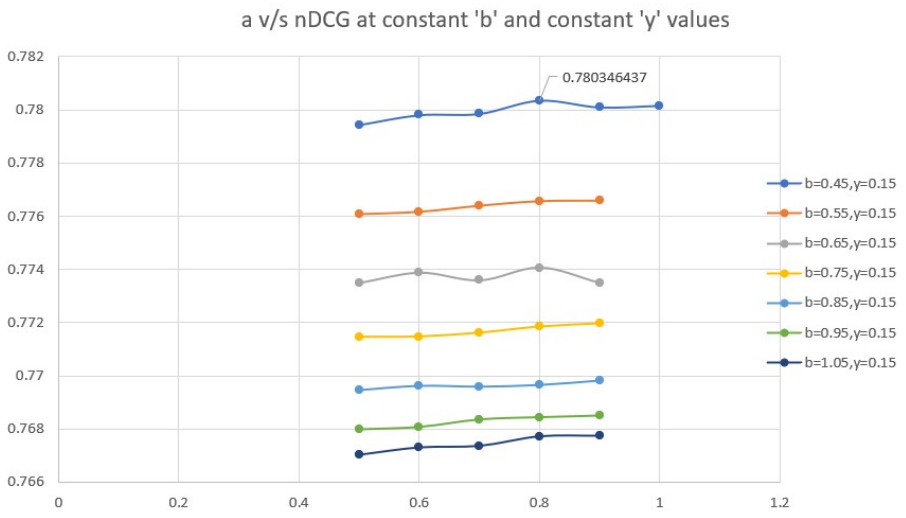

Figure 2: a v/s nDCG score at constant 'b' and constant 'y'

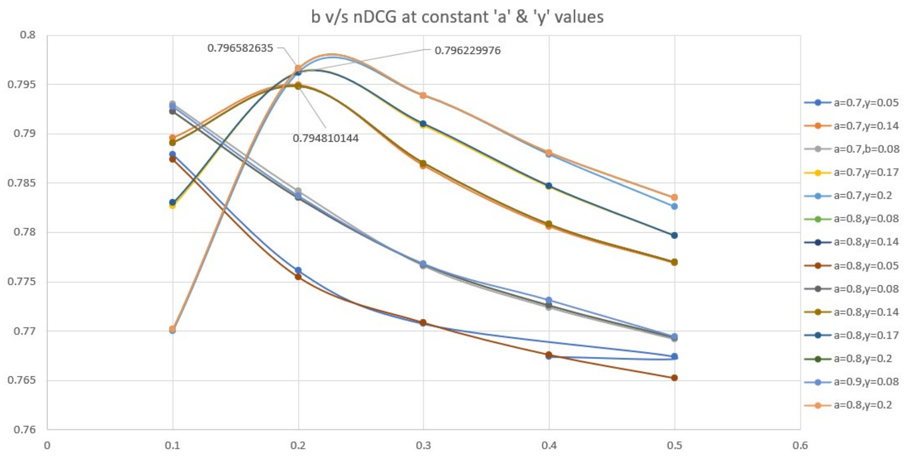

Figure 3: b v/s nDCG score at constant 'a' and constant 'b'

2.3 Part 2: Relevance Model based Language Modeling

For every word in the collection we compute it's probability of being the next word given a relevant query.

$$P(w \mid R) \approx P\left(w \mid q_1 q_2 \ldots q_k\right) \approx \frac{P(w, q_1 q_2 \ldots q_k)}{P(q_1 q_2 \ldots q_k)}$$

Two Methods are used to approximate these probabilities :-

1. **RM1-IID sampling** : In this relevance model we assume that the word $w_i's$ and q_i are mutually independent once we decide on the model M.

$$P\left(w, q_1 \ldots q_k\right) = \sum_{M \in \mathcal{M}} P(M) P\left(w, q_1 \ldots q_k \mid M\right)$$
$$P\left(w, q_1 \ldots q_2 \mid M\right) = P(w \mid M) \prod_{i=1}^{k} P\left(q_i \mid M\right)$$
$$P\left(w, q_1 \ldots q_k\right) = \sum_{M \in \mathcal{M}} P(M) P(w \mid M) \prod_{i=1}^{k} P\left(q_i \mid M\right)$$

2. **RM2-Conditional sampling** : In this relevance model we assume that q_i is independent of w once we decide on model M_i.

$$P\left(w, q_1 \ldots q_k\right) = P(w) \prod_{i=1}^{k} P\left(q_i \mid w\right)$$
$$\mathrm{P}(q_i \mid w) = \sum_{M_i \in \mathcal{M}} P\left(M_i \mid w\right) P\left(q_i \mid M_i\right)$$
$$P\left(w, q_1 \ldots q_k\right) = P(w) \prod \sum_{M_i \in \mathcal{M}} P\left(M_i \mid w\right) P\left(q_i \mid M_i\right)$$
$$P\left(w, q_1 \ldots q_k\right) = P(w) \prod_{i=1}^{k} \sum_{M_i \in \mathcal{M}} P\left(M_i \mid w\right) P\left(q_i \mid M_i\right)$$
$$P(w) = \sum_{M \in \mathcal{M}} P(w \mid M) P(M)$$
$$P\left(M_i \mid w\right) = \frac{P(w \mid M_i) P(M_i)}{P(w)}$$

3. **Implementation Details** :

$$\hat{P}(t \mid M) = \frac{f_{t,d} + \mu \hat{P}_C(t)}{|D| + \mu}$$

   The following assumptions are made in the implementation :

   (a) The entire vocabulary(c) is approximated as the collection of the 100 documents.

   (b) $\mu = 100$

   (c) $P_c(t) = tf_t/100$(collection size)

   (d) $P(M_i) = 1/100$

   (e) M for a query is considered as the top-100 documents retrieved

   (f) All the pre-processing steps are followed as described above for the query and the documents.

2.3.1 Re-Ranking

The cosine similarity of the formulated query is calculated with the top-100 documents and the list is reverse sorted according to the obtained similarity scores.

$\text{sim}\,(d_j, q) = \cos(\theta) = \frac{\overrightarrow{d_j} \cdot \vec{q}}{\left|\vec{d_j}\right| \times |\vec{q}|}$

2.3.2 Query Expansion Terms

The $P(w|M)$ is calculated for all the words in the collection (100 documents retrieved) and sorted according to their probabilities. The top 20 words with the highest probability of occurrence are chosen to expand the query.

2.3.3 Evaluation Metrics

The evaluation scores obtained for the expanded query calculated using treceval is as follows :- are as follows :-

RM1		
	Original Ranking	Re-Ranking
nDCG @ 5	0.4844	0.6008
nDCG @ 10	0.4833	0.5785
nDCG @ 30	0.4613	0.5331
nDCG @ 100	0.4042	0.4237
MAP	0.0576	0.0610

RM2		
	Original Ranking	Re-Ranking
nDCG @ 5	0.4844	0.6105
nDCG @ 10	0.4833	0.5795
nDCG @ 30	0.4613	0.5418
nDCG@ 100	0.4042	0.4244
MAP	0.0576	0.0618

2.3.4 Hyperparameter Tuning

A grid search was performed over a range of μ values.

Highest Average nDCG score was found at μ = (avg doc size)*1.2.

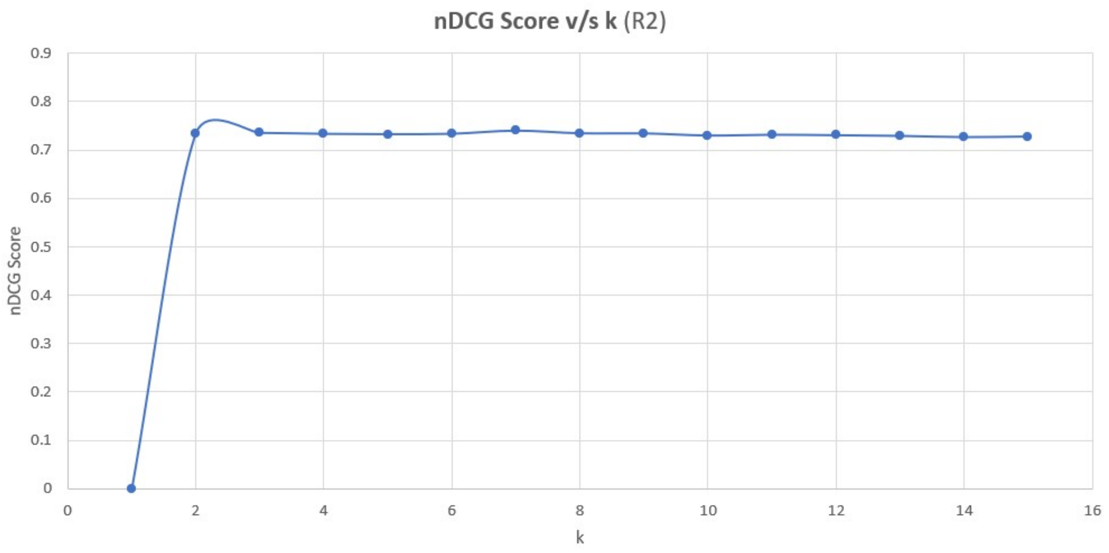

Figure 4: nDCG score v/s k

3 Parameters

Parameter	**Value**
α	0.8
β	0.2
γ	0.2
$P(M_i)$	1/100