Chinmay Mittal (2020CS10336)

Assignment 2 | COL764
Information Retrieval and Web Search

2023-10-21

Document Preprocessing & Language Modeling

I have experimented with the following text-preprocessing techniques:

- **Casing**: Option to convert all text to lowercase or leave as it is.
- **Punctuations**: Option to remove these or keep them as it is.
- **Digits**: Since there is a lot of numerical data, in academic texts and are not directly relevant for answering queries, i have kept an option to replace them by common token $<$ DIGIT $>$
- **Stopwords**: Option to remove stopwords such as i, the etc. (from NLTK)
- **Stemming**: Option to do stemming using the snowball stemmer (from NLTK).

I compute uni-gram, language model probabilities for each document in the top-100 documents retrieved, and also uni-gram probabilities for background model using all documents retrieved collectively.
To handle out of vocabulary terms, some probability mass is reserved for the $<$ UNK $>$ token. Any token in the query or otherwise, not in the documents retrieved are treated as the $<$ UNK $>$ token.
Each of the individial document langauge models are smoothened using Dirichilet smoothing where the background language model is the language model of all the top documents retrieved collectively.

$$P(t|M_j) = \frac{f_{t,d_j} + \mu \hat{P}_c(t)}{|d_j| + \mu}$$

Relevance Based Language Modeling

We take the vocabulary $\mathcal{V}$ to be the all the terms in the initial set of documents retrieved for a query. We then compute uni-gram language models with *Dirichilet* smoothing for each document.
To compute the $P(w|\mathbf{R})$ where $\mathbf{R}$ is the relevance model, we use the following set of equations.

$$p(w|\mathbf{R}) = P(w, q_1, q_2 \ldots q_n)/P(q_1, q_2, \ldots q_n)$$

Here $q_1, q_2 \ldots q_n$ is the Query.

$$P(w, q_1, q_2 \ldots q_n) = \sum_{\mathbf{M} \in \mathcal{M}} P(\mathbf{M}) P(w, q_1, q_2 \ldots q_n|\mathbf{M})$$

Marginalizing over all the document models of the top documents retrieved (Here we assume $P(\mathbf{M})$ is uniform).

$$\Rightarrow P(w, q_1, q_2 \dots q_n) = \sum_{\mathbf{M} \in \mathcal{M}} P(\mathbf{M}) P(w|\mathbf{M}) \Pi_{i=1}^{n} P(q_i|\mathbf{M})$$

Using independence of terms, once the model is chosen.

$$\Rightarrow P(w|\mathbf{R}) = \frac{\sum_{\mathbf{M} \in \mathcal{M}} P(\mathbf{M}) P(w|\mathbf{M}) \Pi_{i=1}^{n} P(q_i|\mathbf{M})}{\sum_{w \in \mathcal{V}} \sum_{\mathbf{M} \in \mathcal{M}} P(\mathbf{M}) P(w|\mathbf{M}) \Pi_{i=1}^{n} P(q_i|\mathbf{M})}$$

Simplifying these equation we get,

$$\Rightarrow P(w|\mathbf{R}) = \frac{\sum_{\mathbf{M} \in \mathcal{M}} P(w|\mathbf{M}) \Pi_{i=1}^{n} P(q_i|\mathbf{M})}{\sum_{\mathbf{M} \in \mathcal{M}} \Pi_{i=1}^{n} P(q_i|\mathbf{M})}$$

We compute these probabilities to estimate the Relevance the model, and re-rank documents by computing the *KL-Divergence* between the Relevance model and the Document Language models using the following equation

$$\mathcal{D}(M_d||M_R) = \sum_{w \in \mathcal{V}} P(w|M_d) log \frac{P(w|M_d)}{P(w|M_R)}$$

Results & Experiments

Effect of Smoothing

Keeping other parameters the same we vary the amount of smoothing (controlled by μ in Dirichlet Smoothing) The Results are as follows:

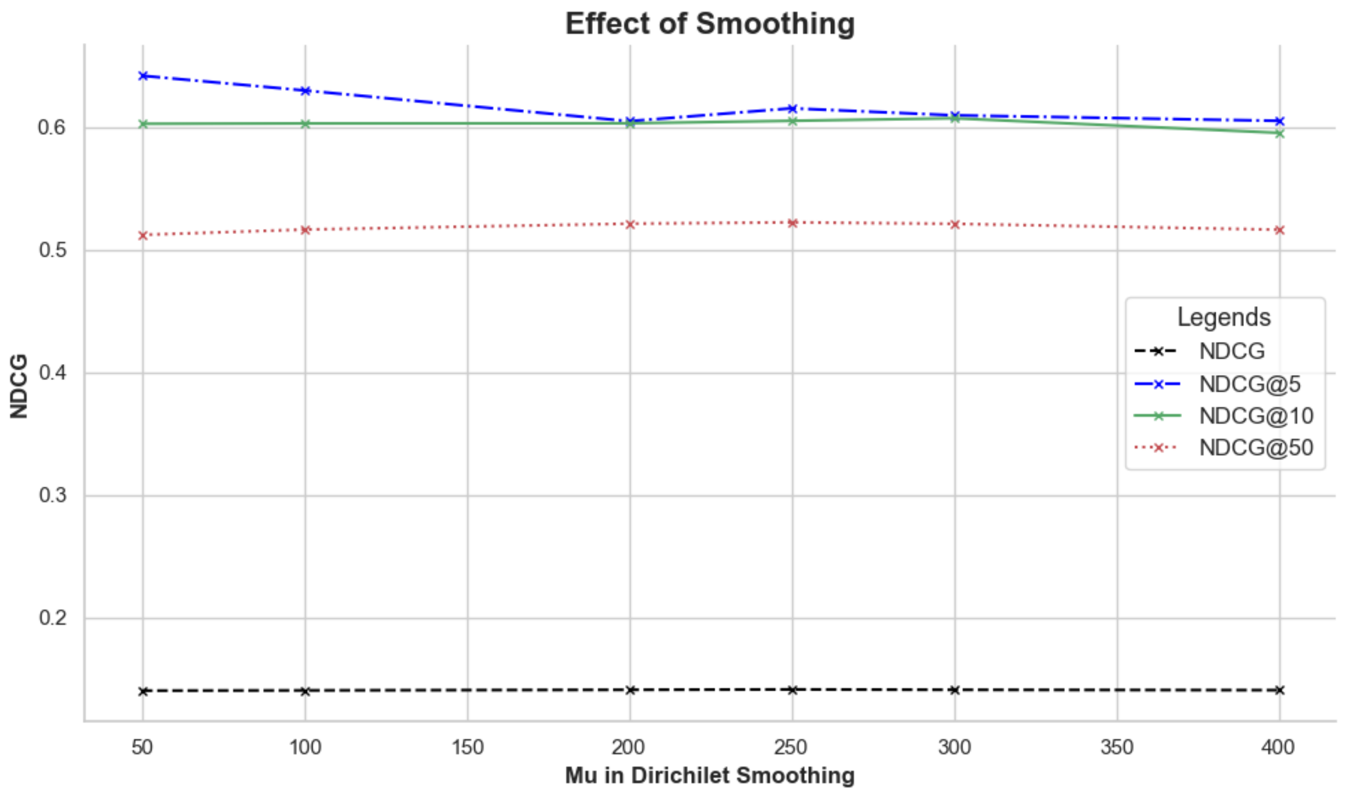


Figure 1: Effect of Smoothing

The best value for μ is around **250**

Effect of Stemming

Keeping others thing same, we see whether stemming helps or not

Parameter	Value	NDCG	NDCG@5	NDCG@10	NDCG@50
Stemming	**True**	**0.1421**	**0.6643**	**0.6232**	0.5201
Stemming	False	0.1415	0.6102	0.6161	**0.5234**

From this experiment we conclude that stemming helps.

Effect of Removing Punctuations

Keeping others thing same, we see whether removing punctuations helps or not.

Parameter	Value	NDCG	NDCG@5	NDCG@10	NDCG@50
Remove Punctuations	**True**	**0.1415**	**0.6102**	**0.6161**	**0.5234**
Remove Punctuations	False	0.1407	0.5970	0.6049	0.5190

From this experiment we conclude that removing punctuations is better.

Effect of Normalizing Digits

Keeping others thing same.

Parameter	**Value**	**NDCG**	**NDCG@5**	**NDCG@10**	**NDCG@50**
Normalizing Digits	True	0.1412	**0.6152**	0.6050	0.5222
Normalizing Digits	**False**	**0.1415**	0.6102	**0.6161**	**0.5234**

From this experiment we conclude that not normalizing digits to the same token is better.

Effect of Stopword Elimination

Keeping others thing same, we see whether or not eliminating stop-words is better.

Parameter	**Value**	**NDCG**	**NDCG@5**	**NDCG@10**	**NDCG@50**
Stopword Elimination	**True**	**0.1415**	**0.6102**	**0.6161**	**0.5234**
Stopword Elimination	False	0.1399	0.6008	0.6003	0.5167

From this we conclude that eliminating stopwords is better.

Effect of Lowercasing

Keeping others thing same, we see whether it is better to convert documents into lowercase or not.

Parameter	**Value**	**NDCG**	**NDCG@5**	**NDCG@10**	**NDCG@50**
Lowercase	**True**	**0.1415**	**0.6102**	**0.6161**	**0.5234**
Lowercase	False	0.1414	0.6093	0.6146	0.5228

From this experiment we conclude that it is better to convert documents to lowercase

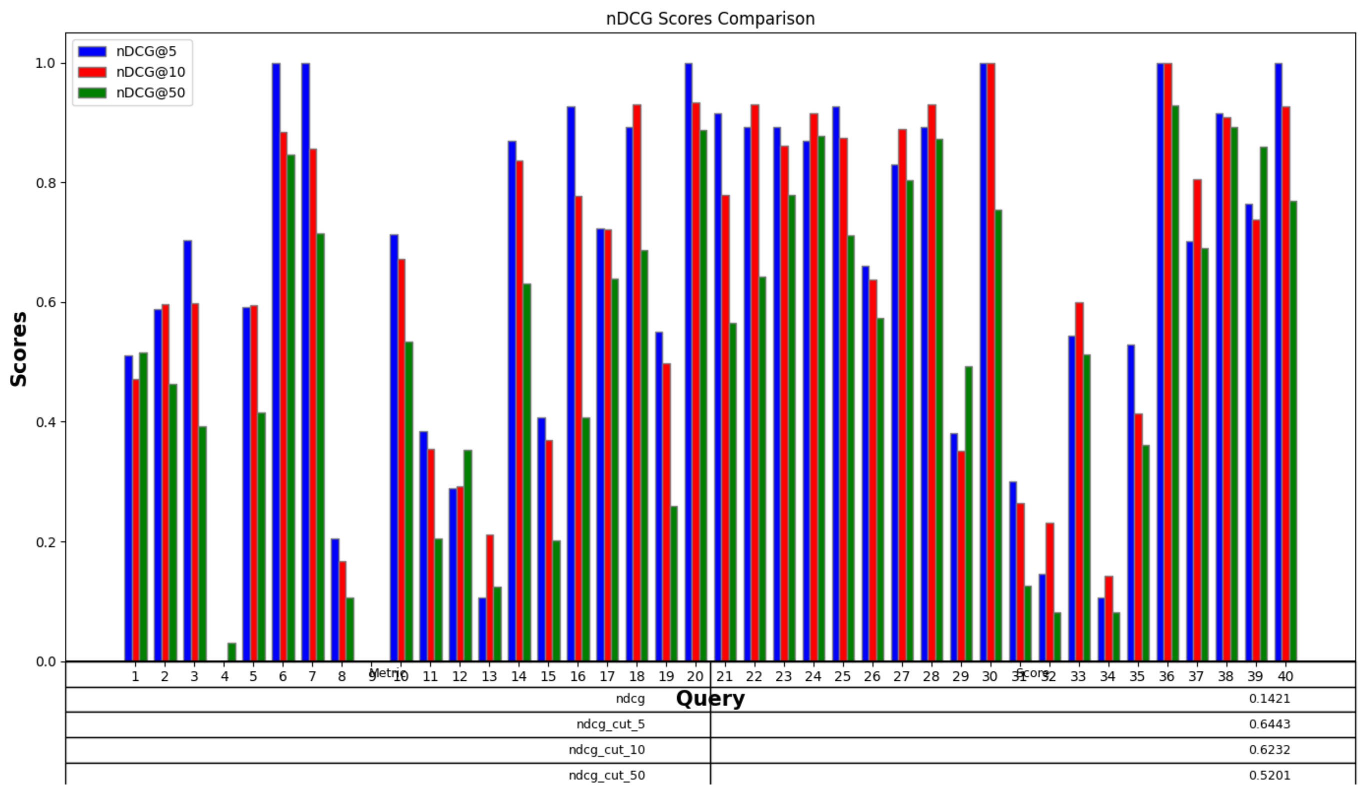
Final Results

Hence the best set of parameters for these experiments are as follows:

Parameter	**Value**
Stopword Removal	True
Normalizing Digits	False
Lowercasing	True
Stemming	True
Smoothing Constant (μ)	250
Removing Punctuations	True

The Results for the same are as follows:

Metric	**Value After Reranking**	**Original Value**
NDCG	0.1421	0.1326
NDCG@5	0.6443	0.4844
NDCG@10	0.6232	0.4833
NDCG@50	0.5201	0.4424



Metric	Scores
ndcg	0.1421
ndcg_cut_5	0.6443
ndcg_cut_10	0.6232
ndcg_cut_50	0.5201

Figure 2: Results for Pseduo Relevance Language Modeling for Each Query

Local Embeddings for Query Expansion

For this task, I first create relevance corpuses, from the top-K documents retrieved for each query. I use this corpus, to train the *Word2Vec* embeddings for each query. This gives a Embeddings matrix $\mathbf{U}$ of size $V \times d$, where V is the size of the vocabulary and d is the dimensions of the embeddings trained.

We then compute $\mathbf{UU^T}$, which is a $V \times V$ matrix, where each term denotes the similarity between two vocabulary terms.

We create the query vector $\mathbf{q}$ of dimension $V \times 1$, where each elements denotes, how many times a vocabulary term is present in the query. If a query term is not in the vocabulary then it is treated as the $<$ *UNK* $>$ token.

We then compute $\mathbf{UU^Tq}$ which is a $V \times 1$ vector denoting the proximity of each term in the vocabulary to the Query Terms. We pick the *Top-K* terms for this, and these are the Query Expansion terms.

We use the weights of the previous multiplication and normalize them to get the Language Model corresponding to the expansion terms, terms which are closer to the query terms are given more weight. This model is called p_{q^+}.

We interpolate, the original query language model (p_q), with this new model, to get the final query language model, according to the following equation.

$$p_{q'}(w) = (1 - \lambda)p_q(w) + \lambda p_{q^+}(w)$$

To re-rank the documents we then use, the *KL-Divergence* between the Document Language models and $p_{q'}$

$$\mathcal{D}(q'||d) = \sum_{w \in \mathcal{V}} p_{q'}(w) log \frac{p_q(d)}{p_d(d)}$$

$\mathcal{D}$ is the scoring function we use for ranking.

Results & Experiments

The following parameters were used from training the Word2Vec Embeddings, these were reused from Google's Implementation without experimentation

Word2Vec Details

Parameter	**Value**
Vector Dimensionalality	100
Size of Context Window	8
Method	Continuos Bag of Words
Training Algorithm	Negative Sampling
Threshold for Downsampling Frequent Words	25

Effect of Number of Terms in Query Expansion

Keeping other things fixed to their best values from previous experiments:

Parameter	**Value**
Stopword Removal	True
Normalizing Digits	False
Lowercasing	True
Stemming	True
Smoothing Constant (μ)	250
Removing Punctuations	True
Expansion Interpolation Constant (λ)	0.5

We vary the number of terms considered in Query Expansion, the Results of which are as follows:

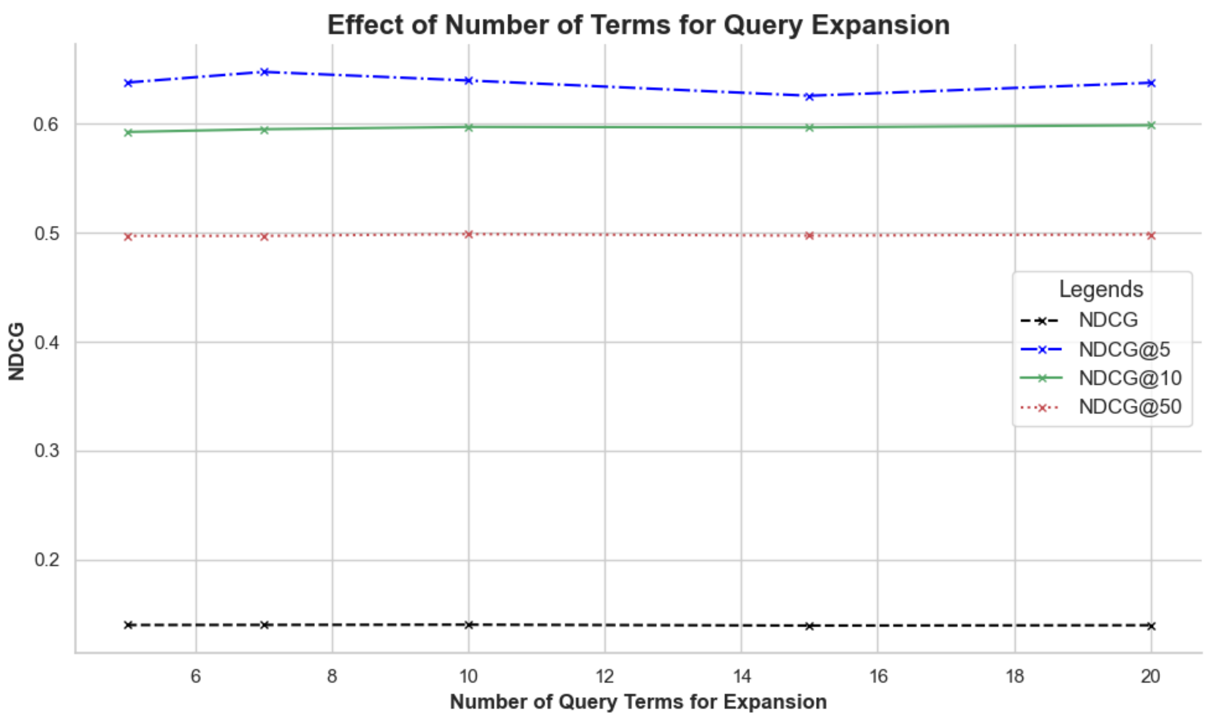


Figure 3: Effect of Number of Query Terms for Expansion

Using this experiment we determine that the best value for number of terms considered for Query Expansion is K=10

Effect of Interpolation Constant (λ)

Keeping other things fixed to their best values from previous experiments:

Parameter	**Value**
Stopword Removal	True
Normalizing Digits	False
Lowercasing	True
Stemming	True
Smoothing Constant (μ)	250
Removing Punctuations	True
Number of Query Terms for Expansion (k)	10

We vary the interpolation constant, which determines the weightage of the expansion terms and the original terms in the Query Language model. The results are as follows:

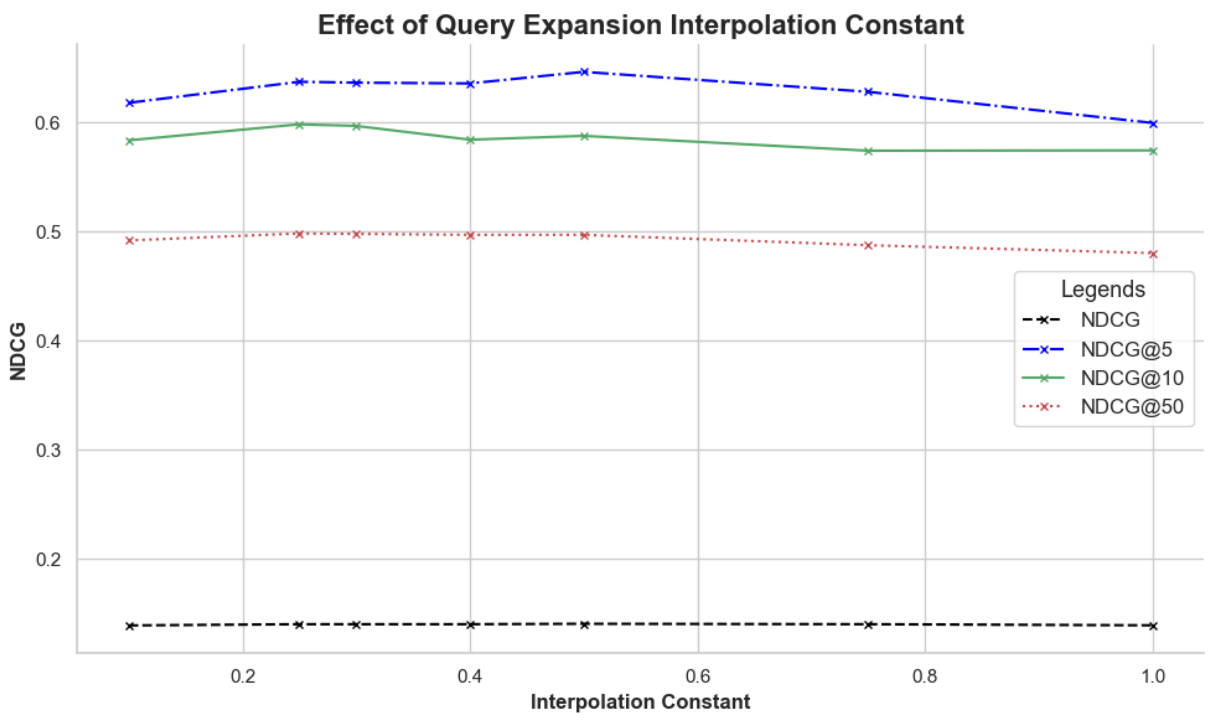


Figure 4: Interpolation Constant λ

Using this experiment we determine that the best value for the interpolation constant (λ) is 0.5

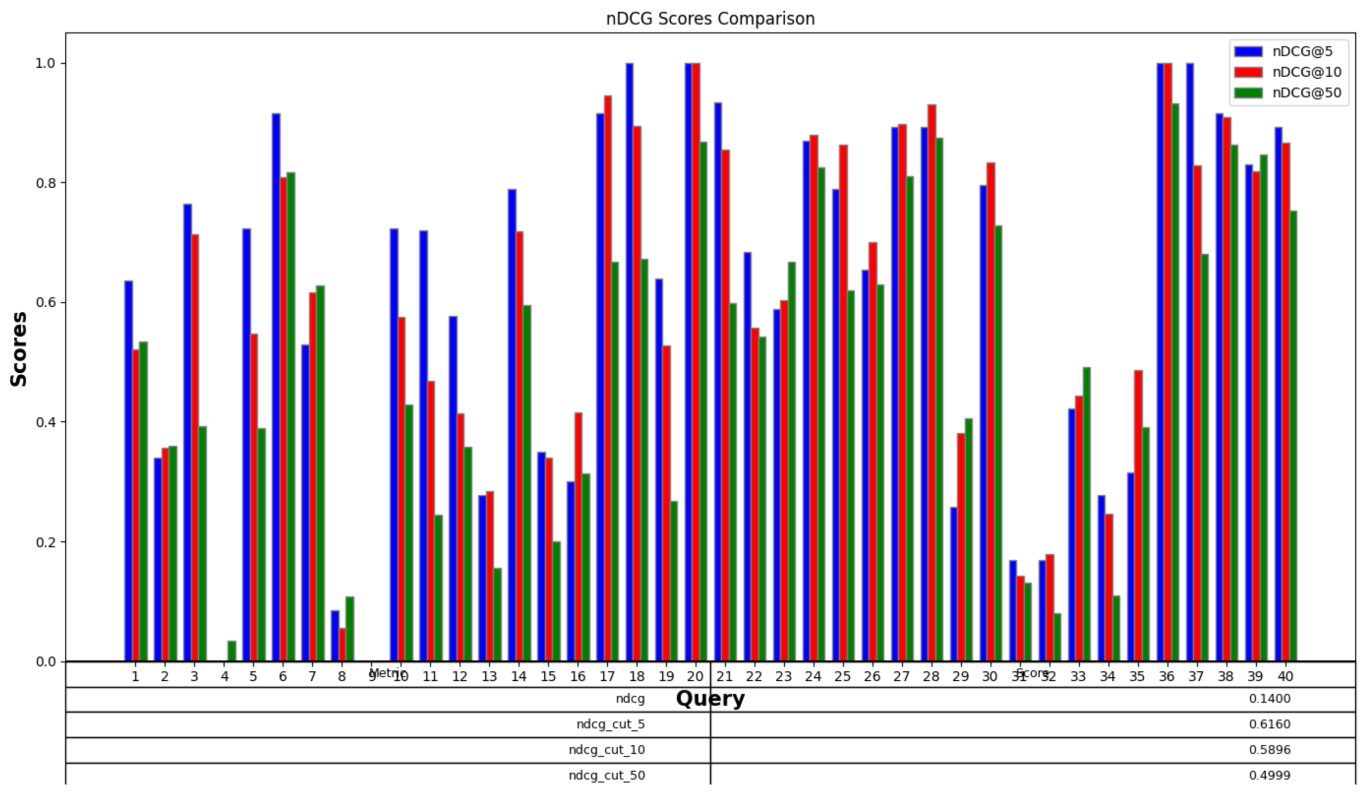
Final Results

Hence the best set of parameters for this experiments are as follows:

Parameter	Value
Stopword Removal	True
Normalizing Digits	False
Lowercasing	True
Stemming	True
Smoothing Constant (μ)	250
Removing Punctuations	True
Number of Query Terms for Expansion (k)	10
Interpolation Constant (λ)	0.5

The Results for the same are as follows:

Metric	Value after Reranking	Original Value
NDCG	0.1400	0.1326
NDCG@5	0.6160	0.4844
NDCG@10	0.5986	0.4833
NDCG@50	0.4999	0.4424



Metric	Score
ndcg	0.1400
ndcg_cut_5	0.6160
ndcg_cut_10	0.5896
ndcg_cut_50	0.4999

Figure 5: Results for Local Embeddings based Expansion for Reranking

Expansion Terms

coronavirus mutations	**coronavirus public datasets**	**coronavirus vaccine candidates**
distinctive	allen	development
variations	investigators	promising
undergone	investigations	preclinical

what alcohol sanitizer kills coronavirus	**masks prevent coronavirus**	**coronavirus outside body**
soap	homemade	sneeze
n-proponal	spread	contagiousness
non-alcohol	nonmedical	enters