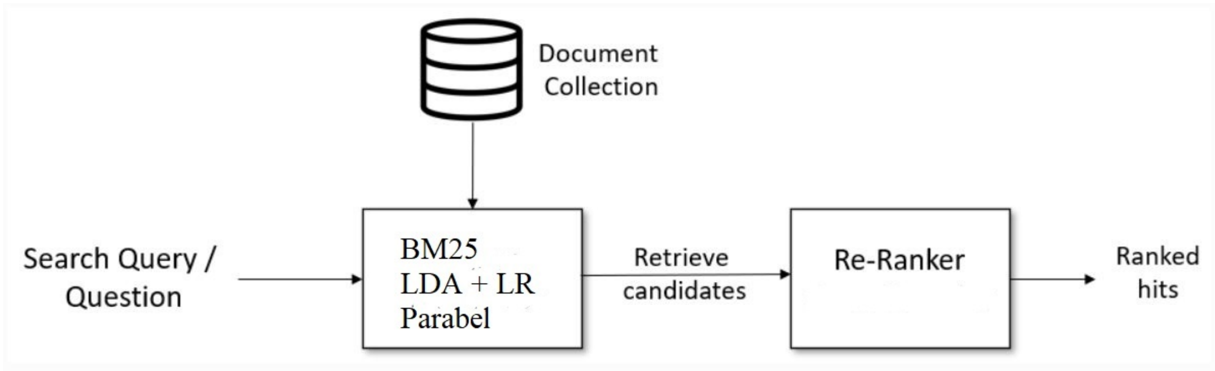
SHIVANGI BITHEL
2020CSY7657
csy207657@cse.iitd.ac.in

Homework Assignment 1

Special Topic in Natural Language Processing

2021-03-10



Given a search query, we first need to use a retrieval system that retrieves a large list of e.g. K possible hits which are potentially relevant for the query (K=10, 50, 100, 500, 1000). For the retrieval, we have three algorithms
1. BM25
2.Parabel
3. LDA + Logistic Regression

There is a possibility that the retrieval system might retrieve documents that are not that relevant for the search query. Hence, in a second stage, we need to use a re-ranker based on a cross-encoder that scores the relevancy of all candidates for the given search query.
The output will be a ranked list of hits we can present to the user.

1 First Exercise

Build the dataset pipeline to feed the training as well as test dataset to the algorithms.

1.1 BM25

BM25 is a probabilistic model that is developed by Stephen E Robertson, Karen Spark Jones, and others. Here we are using another variant of BM25 given by Tortman *et al.* which has proven to be effective in a number of scenarios.

$$rsv_q = \sum_{t \in q} \log \frac{N}{df_t} . \frac{(k_1 + 1).tf_d}{k_1.(1 - b + b.(\frac{L_d}{L_avg})) + tf_d} \tag{1}$$

For a given query q, the retrieval status value, rsv_q, is the sum of individual term, t scores. N is the number of documents in the collection, df_t is the number of the documents containing the term (the document frequency), tf_{td} is the number of times term t occurs in the document d. L_d is the length of the document (in terms) and L_{avg} is the mean of the document lengths. There are two tuning parameters, b, and k_1.
This ATIRE variant of BM25, is using Robertson-Walker *IDF*, which tends to zero as df_t tends to N. This ATIRE function always considers documents containing the query term to be more relevant than those that do not. We are using k_1=1.5, b=0.75 and *epsilon*=0.25 as the value of our tuning parameters. We are creating an indexed corpus using BM25 scores of every document. For every query we are searching this corpus and retrieving the top K documents as relevant documents.

Approach:
Read all D documents from msmarco-docs.tsv
Combine the title and body of the document into a string.
Remove stopwords and punctuations from string and store the tokenized string in a file.
Generate BM25Okapi Model from tokenized documents.
Get scores for every query in the test query file and store in a dictionary.
Sort the scores and generate the output file with top k documents according to scores.

1.2 Parabel

The Parabel algorithm is used for extreme multi-label learning where the objective is to learn classifiers that can annotate each data point with the most relevant subset of labels from an extremely large label set. Parabel is a hybrid 1-vs-All/tree ensemble which efficiently learns a balanced label hierarchy from training data so as to cut overall training and prediction costs from linear to logarithmic in the number of labels. It then learns a probabilistic model based on the learnt label hierarchy for the joint label distribution conditioned on a data point. This model generalizes the multi-class hierarchical softmax model used in language modelling and word-vector embeddings to the multi-label setting. Finally, Parabel develops efficient log-time training and prediction algorithms based on the proposed probabilistic model.

Approach:
Read all D documents from msmarco-docs.tsv
Combine the title and body of the document into a string.
Remove stopwords and punctuations from string and store the tokenized string in a file.
Create dictionary and document-term matrix from tokenized documents.
Generate Bag-of-word feature dataset from dictionary .
Using the above dataset files run Parabel model.
Using the Model, generate result for test data.

Dataset:
The data set to be generated for Parabel should be of the form:
Header Line: TotalPoints NumFeatures NumLabels
1 line per datapoint : label1,label2,...labelk ft1:ft1val ft2:ft2val ft3:ft3val .. ftd:ftdval

For Example:
3809 5000 3993
3,4,52,260,518,992,1543 0:0.101468 1:0.554374 2:0.235760 3:0.065255 8:0.152305 10:0.155051 11:0.182930 13:0.213172 14:2.387193 15:0.274654 16:0.320825 17:7.357962 18:0.703709 21:0.387763 22:0.395067

Here first argument is total number of points i.e. Total number of documents
Second argument is total number of features i.e. Size of vocabulary
Third argument is number of labels i.e. Number of Queries for our case
For each data point, i.e. a document we first right all relevant labels(queries) to that document.
For every word of document, we represent it as a index from data dictionary along with either the count of the word in corpus or its TF-IDF score.

1.3 LDA + Logistic Regression

Latent Dirichlet Allocation:
Latent Dirichlet Allocation is a well-known topic modeling algorithm that infers semantic structure from text data, and ultimately helps answer the question on ”what is this document about?”. It can be used to featurize any text fields as low-dimensional topical vectors.

Approach:
Read all D documents from msmarco-docs.tsv
Combine the title and body of the document into a string.
Remove stopwords and punctuations from string and store the tokenized string in a file.
Create dictionary and document-term-matrix from tokenized documents.
Generate LDA Model using gensim.models.ldamulticore
To decide on number of topics there are various approaches:
1. Run LDA on corpus with different numbers of topics and see if word distribution per topic seems sensible.
2. Examine the coherence scores of LDA model, and effectively grid search to choose the highest coherence.
3.Create some of LDA models with different topic values, then see how these perform in the supervised classification model training.
As Running multiple LDA models on the dataset as huge as ours and then try different numbers seems difficult. I referred the paper **Learning to Classify Short and Sparse Text Web with Hidden Topics from Large-scale Data Collections**
I chose the value as 150.
Now that we have the LDA model, we have to generate topic vectors for each of the topics.

Logistic Regression:
In natural language processing, logistic regression is the baseline supervised machine learning algorithm for classification, and also has a very close relationship with neural networks. Logistic regression can be used to classify an observation into one of two classes (like ‘positive sentiment’ and ‘negative sentiment’), or into one of many classes.

Approach:
As we have the number of topics with us, we will now train a one vs all classifier for each of the topic.
For a new query, first we will use trained LDA model model to generate its topic distribution and predict its topic from all 150 topics.
Then we will pick top K documents belonging to the topic which is same to the topic of query.

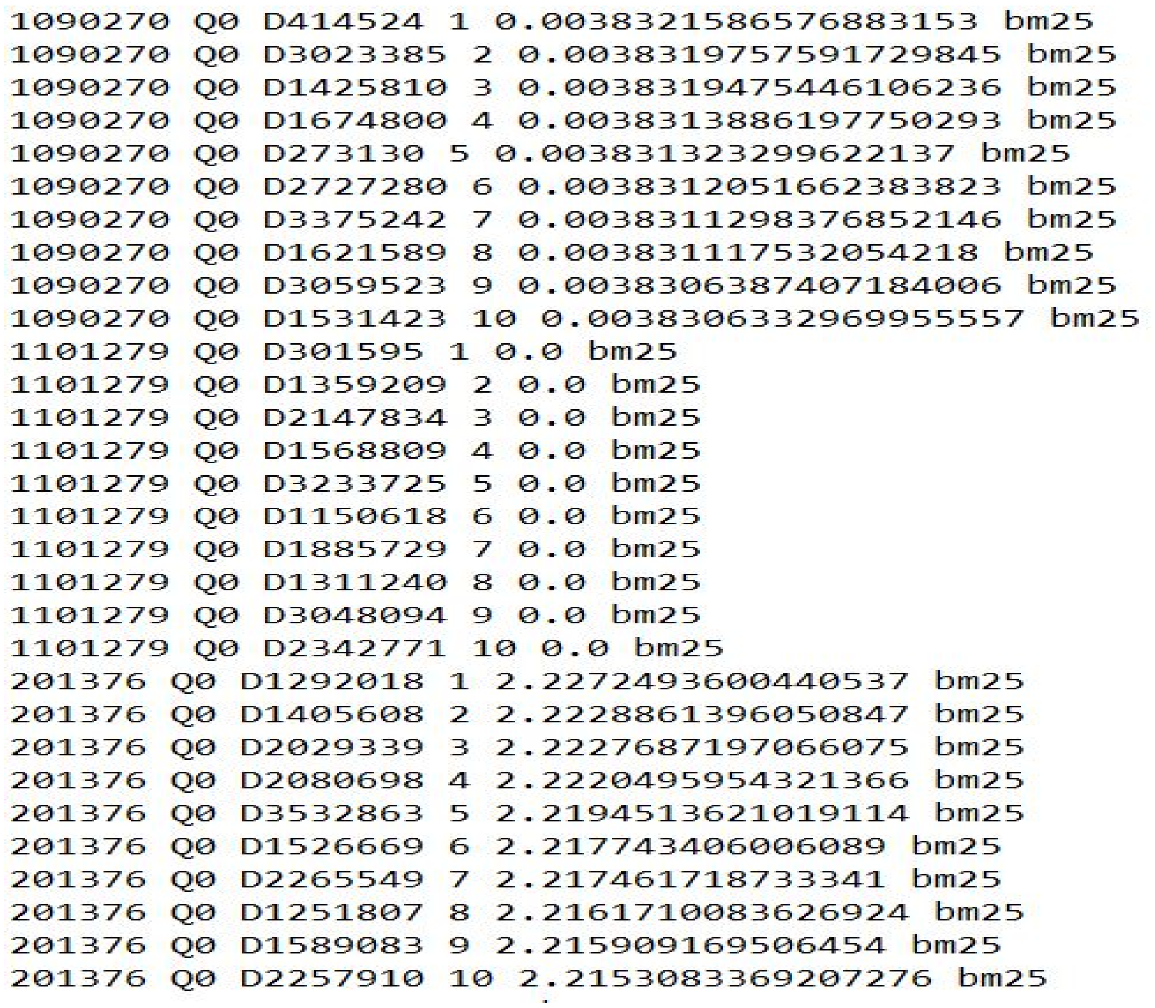
2 Second Exercise

Pseudo relevance feedback (PRF)

For this task we were asked to use Query Expansion techniques to improve the ranking of retrieved documents. We can use different techniques to expand the query like:
1. Using Synonyms from wordnet
2. Language Modelling Techniques

Here I was trying Cross - encoder technique to re-rank the documents but could not complete that also because of some issue with the requirements.

3 Results



```
1090270 Q0 D414524 1 0.003832158657688315З bm25
1090270 Q0 D3023385 2 0.0038319757591729845 bm25
1090270 Q0 D1425810 3 0.0038319475446106236 bm25
1090270 Q0 D1674800 4 0.0038313886197750293 bm25
1090270 Q0 D273130 5 0.003831323299622137 bm25
1090270 Q0 D2727280 6 0.0038312051662383823 bm25
1090270 Q0 D3375242 7 0.0038311298376852146 bm25
1090270 Q0 D1621589 8 0.003831117532054218 bm25
1090270 Q0 D3059523 9 0.0038306387407184006 bm25
1090270 Q0 D1531423 10 0.0038306332969955557 bm25
1101279 Q0 D301595 1 0.0 bm25
1101279 Q0 D1359209 2 0.0 bm25
1101279 Q0 D2147834 3 0.0 bm25
1101279 Q0 D1568809 4 0.0 bm25
1101279 Q0 D3233725 5 0.0 bm25
1101279 Q0 D1150618 6 0.0 bm25
1101279 Q0 D1885729 7 0.0 bm25
1101279 Q0 D1311240 8 0.0 bm25
1101279 Q0 D3048094 9 0.0 bm25
1101279 Q0 D2342771 10 0.0 bm25
201376 Q0 D1292018 1 2.2272493600440537 bm25
201376 Q0 D1405608 2 2.2228861396050847 bm25
201376 Q0 D2029339 3 2.2227687197066075 bm25
201376 Q0 D2080698 4 2.2220495954321366 bm25
201376 Q0 D3532863 5 2.2194513621019114 bm25
201376 Q0 D1526669 6 2.217743406006089 bm25
201376 Q0 D2265549 7 2.217461718733341 bm25
201376 Q0 D1251807 8 2.2161710083626924 bm25
201376 Q0 D1589083 9 2.215909169506454 bm25
201376 Q0 D2257910 10 2.2153083369207276 bm25
```

The output file of BM25 looks like the above image.

First argument is the Query Id.
Second argument is the iterator.
Third argument is the Document Id.
Fourth argument is the rank.
Fifth argument is the BM25 score.
Sixth argument is the name of algorithm ”bm25”

As you can observe from the figure, some of the Query - Document is getting rank 0 here. The number of such Query- Document pair is quite large in the complete output file, which is making the metrics value as 0.

Possible reason for such results
As the document number is very large and thus the size of vocabulary is also quite large which is making the Query vectors very sparse. Similar comment is made in the paper **Dense Passage Retrieval for Open-Domain Question Answering**.

4 References

1.Improvements to BM25 and Language Models Examined
(ADCS ’14: Proceedings of the 2014 Australasian Document Computing Symposium)

2. Document Classification by Topic Labeling
(SIGIR ’13: Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval)

3. Dense Passage Retrieval for Open-Domain Question Answering (EMNLP 2020)

4. Sentence Transformers Documentation

5.Speech and Language Processing by Daniel Jurafsky, James H. Martin

6. Using LDA Topic Models as a Classification Model Input

7.Parabel: Partitioned Label Trees for Extreme Classification with Application to Dynamic Search Advertising
(WWW ’18: Proceedings of the 2018 World Wide Web Conference)

8. https://pypi.org/project/rank-bm25/

9. Learning to classify short and sparse text web with hidden topics from large-scale data collections
(WWW ’08: Proceedings of the 17th international conference on World Wide Web)