Information Retrieval and Web Search

Assignment 2

Rajat Singh (csz208507)

1) Probabilistic Retrieval Query expansion

In this part of the assignment, we are trying to implement the probabilistic retrieval Query expansion. In this we need to add a word in the query after every iteration.

In this assignment we are given a list of 100 documents relative to each query using BM25 now our task is to retrieve those documents from the corpus and process our query.

To achieve fast retrieval of the document from the corpus I have precomputed a dictionary in memory which stores the offset of the given document id. So now my retrieval of the document is O (1) operation if I know the offset. This saves the time to again and again traverse the corpus to get the details of the 100 documents that are relevant to the given query.

After retrieval of the 100 top documents next step was to compute term weight of each term in the relevant documents. I took the hypothesis that top 10 documents are marked as relevant, so I have computed the weight of those terms who lies in the top 10 documents and sorted according the weight and the word whose weight is the maximum of all is added to the query and then we computed the weight of all the 100 documents for the new query using BM25 and repeat the process given number of times

Number of times query got expanded	1	2	3	4	5	6	7	8	9	10
ndcg	0.2711	0.2597	0.2551	0.2500	0.2475	0.2429	0.2412	0.2391	0.2357	0.2323

The ndcg score is continuously decreasing because we are choosing top 10 documents as the relevant documents which is not the case in real.

Command for Probabilistic retrieval query expansion:

```
$ python3 prob_rerank.py <query-file> <top-100-file> <collection-file> <expansion-limit>
```

2) Relevance Model based Language Modeling

A) Unigram Model with Dirichlet Smoothing

In this part of the assignment, we are applying language model to calculate the weight of the document for a given query using unigram model with Dirichlet smoothing. In unigram model I took a hypothesis that the corpus shrinks to the top 100 documents that is listed in "top100-file" for every query using BM25.

For this also I have pre-computed the corpus and find the offset for all the document ids and stored in a dictionary. After computing the offset, I pick the top 100 documents and calculate the length of each document and the frequency of each term in that document and store it in the dictionary.

After getting all the data structures we are using the given below equation to calculate the weight for each document for a given query.

$$\log(P(q|d)) = \sum_{i=1}^{n} \log \frac{f_{t_i,d} + \mu \hat{P}_C(t_i)}{|D| + \mu}$$

Where

$$\hat{P}_C(t) = \frac{f_{t,C}}{L_C}$$

And here my corpus is 100 documents. After getting the score of each document I have sorted the documents with the score and store in the file.

My ndcg score for Unigram Model with Dirichlet Smoothing: 0.3439

Command for Unigram Model with Dirichlet Smoothing:

$ python3 lm_rerank.py [query-file] [top-100-file] [collection-file] [model=uni]

B) Bigram Model with Dirichlet Smoothing with Unigram Backoff

In this part of the assignment, we are applying language model to calculate the weight of the document for a given query using bigram model with Dirichlet smoothing with unigram backoff. Unlike unigram I have paired two words together with space in the document and then try to calculate the weight of the document for the given query.

I have used the following equation to calculate the score of the document

Bigram: $$P(w_i|w_{i-1})=\lambda_2 P_{ML}(w_i|w_{i-1})+(1-\lambda_2)P(w_i)$$

Unigram: $$P(w_i)=\lambda_1 P_{ML}(w_i)+(1-\lambda_1)\frac{1}{N}$$

8

Where

$$\lambda=\frac{|d|}{|d|+\mu}$$

So, for every query I paired the words and calculated the score of the documents for that word and then sort the document and store it into the file.

Command for bigram Model with Dirichlet Smoothing with unigram back off

```
$ python3 lm_rerank.py [query-file] [top-100-file] [collection-file] [model=bi]
```