Passage Retrieval Report

Anonymous ACL submission

1 Introduction

Information retrieval refers to the process and technology of organizing information in a certain way and finding relevant information according to the needs of information users. The project solves the problem of passages retrieval by implementing tf-idf, bm25 and likelihood language model with three smoothing methods. This report will answer the questions in each task, show the partial running results and introduce the code implementation ideas.

2 Task 1

'Nltk' python package provides a variety of Application Program Interfaces (APIs) for text processing and it is used in this program for data cleaning. Tokenization, change the string to lower case, remove punctuation and lemmatization are performed in sequence. Stemming is not used because using this method would decrease nearly thirty thousand terms in vocabulary formed.

After implementing the word count method, one python dict is constructed which stores every unique terms with their number of occurrence. These items are sorted in a descending order (from high to low), and in this way the figure of normalized frequency against ranking can be plotted.

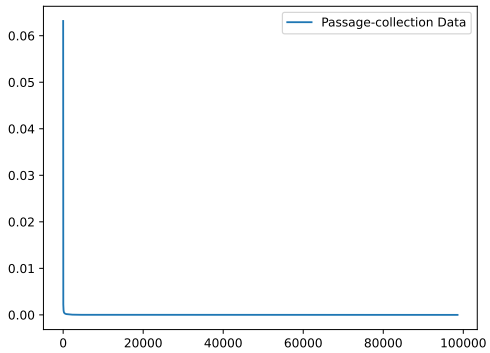

Figure 1: Normalized Frequency against Frequency Ranking

As it shown above, the shape of the curve in the figure 1 is quite similar with the curve formed by Zipf's law, which is introduced in lecture courseware. Therefore, terms in the corpus qualitatively follow Zipf's law.

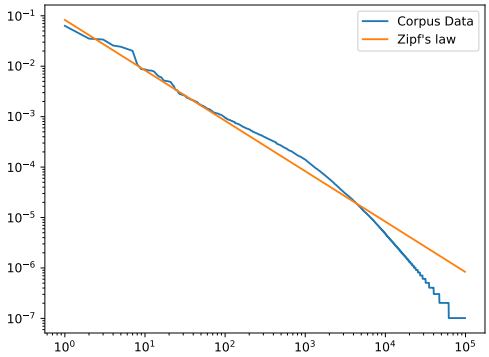

Figure 2: Normalized Frequency against Frequency Ranking (log-log plot)

With the x-coordinate and y-coordinate data used in figure 1, log-log plot can also be generated, which is shown in figure 2. It claims that the trends of the two curves are roughly the same. However, there is a significant drop and vibration at the tail of the actual corpus curve. It is thought that this is because these particularly low-ranked terms appear too infrequently in the corpus.

3 Task 2

The approach of generating the inverted index in this program is listed below:

1. Read the csv file and then get the dataframe of 'candidate-passages-top1000.tsv'.

2. Extract the 'pid', 'passage' columns from the dataframe obtained in step 1 and then remove the duplication. In this way, the dataframe of 'pid' corresponding to the 'passage' one by one is established.

3. Remove the stop words in vocabulary obtained in task 1 as the key of the inverted index and then initialize it as an empty list.

4. Iterate over each row in the dataframe obtained in step 2, divide the passage into a list of words, and count the number of occurrences of each unique term in the passage. Then for each term, if it appears in the key of the inverted index, the 'pid' corresponding to the passage and the number of occurrences of that term are added into the list initialized in step 3.

In summary, the information stored for each term in the inverted index is a list of tuples which contains the 'pid' and number of occurrences of the term in this passage. In this way, tf-idf value for a term in a specific passage can be easily computed.

4 Task 3

The tf-idf vector representations of all passages are extremely large, the dimension is 182469 x 98319, which is difficult to store in local computer. Therefore, a function to compute the tf-idf vector representation of one passage is implemented in task 3. Users need to pass in the 'pid' parameter to get the corresponding result. The basic idea is to traverse each word of this passage, calculate the word frequency in this passage and multiply its 'idf' value. Use the product as the result of the dimension where the word is located.

The dimension of tf-idf vector representation of all queries is 200 x 98319, which can be stored in local computer. The basic idea is to loop the function introduced above.

5 Task 4

From the lecture, we know that the best smoothing will not smooth too much and not make a lot of assumption. Basically, it seems that Dirichlet smoothing finds a balance between these two criteria. Furthermore, Dirichlet smoothing performs well when the length of query is small, which consistent with the data set of this project. Therefore, it is expected that Dirichlet smoothing language model will work better.

Query likelihood language model with Laplace smoothing and Lidstone correction with $\epsilon = 0.1$ are expected to be more similar.

$$f(x, y) = \frac{tf_{W,D} + 1}{|D| + |V|} \tag{1}$$

$$f(x, y) = \frac{tf_{W,D} + \epsilon}{|D| + \epsilon\,|V|} \tag{2}$$

As it shown above, the equation 1 represents the Laplace smoothing while the equation 2 represents the other one. In the corpus that used in this project, the lengths of query and passage are relatively small. The average length of a passage is only about 50 words. The number of passports is 180,000. That is, $|D|$ in the formula is very small compared to $|V|$ and can be ignored. This means that when $tf_{W,D}$ is 0, the results of the two formulas are basically the same.

I think the value $\epsilon = 0.1$ in the Lidstone correction is not a good choice. Since the Lidstone correction is used to decrease the weight to unseen terms, but in fact the result will increase a lot.

If we set $\mu = 5000$ in Dirichlet smoothing, there will be more weight given to the probability of the whole corpus to produce the query, and correspondingly the weight of the probability of the specific passage to produce the query will decrease. Therefore, it will not be a appropriate way to perform this.