Traditional Information Retrieve: An Experiment

Anonymous ACL submission

Abstract

To put the learned knowledge into practice and finish the project, traditional information retrieval should be studied, including the TF-IDF-based, BM25, and other smooth methods. Before implementing this algorithm, the term frequency should be counted and analyzed. In addition, the inverted index should be constructed correctly to provide the inverse document frequency and the related information. Then, it should apply cosine similarity to compare each query to each passage after implementing the specific algorithm. Ultimately, all algorithm is analyzed based on the requirements of the project.

1 The Term-Based Analysis

1.1 The Data Preprocessing

The stopwords removal is not limited to this problem, it needs other preprocessing steps. In this problem, the special symbols removal and document case unification are applied.

- The special symbol removal: It should remove special symbols, such as commas, periods, ellipses, and other special symbols that affect regular expressions. This is because these symbols are not the terms, and they will induce the unnecessary error in the following problem using the regular expressions that need to be eliminated;

- The document case unification: It should transfer all letters in the document into the lower case because in Python if the capitalization is not uniform, this will lead to different capitalizations of some of the same words will be recognized as different words, and affect the statistical accuracy.

After implementing the data preprocessing step, it obtains 189046 unique words in total.

1.2 The Zipf's Law Analysis

Zipf's law denotes that in a natural language corpus, the frequency of a word is inversely proportional to its rank in the frequency table(Linders and Louwerse, 2022). However, because the term frequency is large in a corpus so that it may induce the computational complex problem, it needs to normalize the term frequency f by Zipfian distribution ($s = 1$)(Lavi-Rotbain and Arnon, 2022):

$$f(k; s, N) = \frac{k^{-s}}{\sum_{i=1}^{N} i^{-s}} \quad (1)$$

where k denotes the term frequency rank of the given word. After normalizing the term frequency, it can obtain the Top-50 words shown in Fig.1.

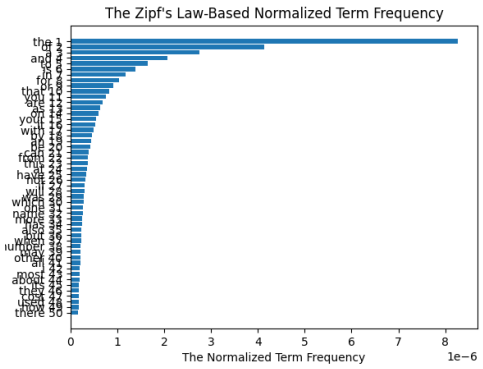

Figure 1: The Normalized Term Frequency

From the figure, it can be discovered that the stopwords place almost Top-50 because they have not been eliminated. In addition, it can be roughly observed that all words obey Zip's law. To quantize the trend, it draws the log-log plot based on the normalized term frequency and the real Zipf's law in Fig.2.

From this plot, it can know that the trend of the predicted distribution largely obeys the real Zip's law because the plot is almost a straight line. However, on words with low word frequency, it can be seen that the line is slightly curved.

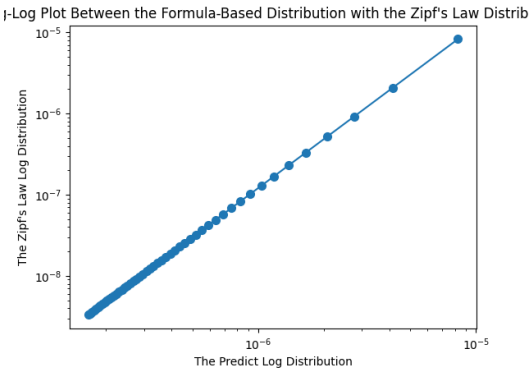

Figure 2: Log-Log Plot Between the Formula-Based Distribution with the Zipf's Law Distribution

2 Inverted index Construction

2.1 Data Preprocessing

Notice that in this problem, it can remove the stopwords to help to retrieve the information. Therefore, it can add the stopwords removal into the data preprocessing step. However, the NLTK library is not allowed to be used, the normalized term frequency should be applied to extract the stopwords - simply delete the Top-50 words because the stopwords occur considerably more than other words and the number of the true stopwords is mainly 50-100. Therefore, deleting the words is reasonable.

In addition, since in a given file, a passage may correspond to multiple queries, these duplicate passages should be deleted according to the requirements of the title. This can also be used as part of preprocessing and is reasonable due to the requirements of the problem.

2.2 Information Storage in the Inverted index

As for the information storage, it only preserves the passage ID to the values and only the word itself as the key of the inverted index. This is because there is no repeat passage and the ID can represent the passage, which indicates that the ID can represent all information for a certain passage, which makes sense.

In addition, the significance of constructing the inverted index is to obtain the inverse document frequency (IDF) in the TF-IDF-based model and other models. Therefore, it remains necessary for constructing another inverted index for the queries and storing the query ID correspondingly.

2.3 Inverted index Construction Algorithm

The method to construct the index is to simply use the "for" loop. The first loop is to obtain one passage or one query. Then, the second loop is to enumerate all words in the passage and add the passage ID or query ID into the inverted index.

Although the time complexity is high, the real execution time can be controlled in a reasonable time in practice.

3 TF-IDF-Based Model

3.1 Data Preprocessing

The step of the data preprocessing is the same as the the problem mentioned before. Therefore, in the preceding problem, it would not mentioned the data preprocessing step.

3.2 TF-IDF Algorithm for The Word

The term frequency (TF) is extracted in the previous chapter, which can be obtained directly for each word. The intervened document frequency can be obtained from the previous chapter also. However, they should be installed correct for the TF-IDF-based formula:

$$V_i = TF_i \times ln\frac{N_D}{1 + N_C} \tag{2}$$

where N_D denotes the total document number, N_C denotes the number of documents in which the word appears, which can be obtained directly by calculating the length of the document ID based on the key of the word in the inverted index.

3.3 TF-IDF Algorithm for The Sentence

Considering that only a few words appear in a passage, it is very wasteful to store the TF-IDF values of words in the form of a list, especially when the corpus has many passages and queries. Therefore, dictionary storage can be used. For a passage and query, only the words that appear are stored, which can save a lot of space and have great benefits for the cosine similarity described below.

3.4 Set-Based Cosine Similarity Algorithm

To calculate the cosine similarity, it can follow the formula:

$$S(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}| \cdot |\mathbf{b}|} \tag{3}$$

Where the length of the vector a, b is $|\mathbf{a}|$ and $|\mathbf{b}|$ respectively. In this equation, the length of the vector only relies on the non-zero element and is not sensitive to the order of those elements because according to the definition, the length of the vector can be written as:

$$|\mathbf{a}| = \sqrt{\sum a_i^2} \tag{4}$$

Therefore, if $a_i = 0$, the sum will not be changed. In addition, according to the exchange rate of the sum, it can be proved that:

$$a_i^2 + a_j^2 = a_j^2 + a_i^2 \tag{5}$$

Thus the length of the vector is not sensitive to the order of the word. Consequently, it can calculate the length directly by calculating the sum of the square of the values of the TF-IDF dictionary of a sentence.

Similarly, it can only calculate the sum of the squares of the words between the corresponding query and passage as the numerator because the numerator can be written as:

$$\mathbf{a} \cdot \mathbf{b} = \sum a_i b_i \tag{6}$$

Therefore, if $a_i = 0$ (denotes the word never occurs in sentence A) or $b_i = 0$ (denotes the word never occurs in sentence B), it will not have any influence on the value of the numerator.

However, if $a_i \neq 0$ and $b_i \neq 0$, the value of the numerator would be not zero. In this case, only the intersection of the query and passage takes part in the calculation. In addition, because TF-IDF and other traditional information retrieval methods have no order representation, it is easy to assign $a_i = b_i$ in this equation.

By the above simplification, it can calculate the similarity between the query and the passage fastly.

4 BM25 Algorithm

The BM25 is the improved version of the TF-IDF and can obtain the similarity directly (not based on the cosine similarity) (Kadhim, 2019). In detail, the formula can be written as:

$$S(Q, P) = \sum_{t \in Q} P_1 \cdot P_2 \cdot P_3 \tag{7}$$

Where each part can be expressed as:

$$\begin{cases} P_1 = ln \frac{N_D}{1+N_C} \\ P_2 = \frac{(k_1+1)TF_{td}}{k_1[(1-b)+b\times\frac{L_d}{L_{avg}}]+TF_{td}} \\ P_3 = \frac{(k_2+1)TF_{td}}{k_3+TF_{td}} \end{cases} \tag{8}$$

Where TF_{td} denotes that the term frequency of the word t in the given document, k_1, b, k_2 are the adjustable parameters, which are given by the problem. The L_d denotes the length of the document and L_{avg} denotes the average document length.

By using the "for" loop, it can substitute the word in each query and each passage to obtain the similarity directly.

5 Query Likelihood-Based Algorithm

5.1 Baseline Model

The baseline model of the query likelihood algorithm can be expressed as:

$$\hat{p}(Q|D) = \prod_{t \in Q} \frac{TF_{td}}{L_d} \tag{9}$$

From this, it can infer that it can also use the "for" loop to calculate the model and then calculate the natural logarithm of the probability according to the requirement of the problem.

However, the problem requires applying the smooth-based method, including the laplacian method, the Lidstone method, and the Dirichlet method.

5.2 Laplacian Smooth Method

The method adds "1" to all term frequencies to avoid zero value and prevent the denominator from becoming 0. In addition, it can help to estimate the unknown probability of the word. The operation is easy, and only uses one line of code.

5.3 Lidstone Smooth Method

The Lidstone method is like the laplacian method, which adds ϵ to all term frequency rather than "1", which can show its adaptation.

5.4 Dirichlet Smooth Method

The Dirichlet smooth method changes the formula of the baseline model, which adds a parameter μ in the denominator, which can be expressed as:

$$\hat{p}(Q|D) = \prod_{t \in Q} \frac{TF_{td} + \mu TF_t}{L_d + \mu} \tag{10}$$

Notice that the parameter μ still occurs in the molecular, with the multiplier of the overall term frequency of the word t. It can help to extract the characteristic of a certain word in the corpus and improve the accuracy.

5.5 Parameter Analysis

Based on the analysis above, can summarize 2 different parameters. For the parameter ϵ in the Lidstone smooth method, it can be observed that the smaller the value of this parameter is, the larger the corresponding value is when the denominator is calculated as 0 and this value is added. As it approaches 0, the obtained value is infinite. It can be seen that the smaller the value, the worse the smoothing effect, and may affect the overall value. Therefore, the value of $\epsilon = 0.1$ is not good, and a larger value needs to be taken to ensure the smoothing effect.

For the parameter μ, it can be witnessed that The smaller the value of this parameter, the smoothing effect can refer to as Lidstone smoothing. If the value of this parameter is too large, it may also cause excessive smoothing, so that the word frequency calculated in the passage corresponding to each query is almost the same. This is because the value of TF_t is too high. Information loss caused by large. To sum up, the value of $\mu = 5000$ is inappropriate and should be lowered.

References

Ammar Ismael Kadhim. 2019. Term weighting for feature extraction on twitter: A comparison between bm25 and tf-idf. In *2019 international conference on advanced science and engineering (ICOASE)*, pages 124–128. IEEE.

Ori Lavi-Rotbain and Inbal Arnon. 2022. The learnability consequences of zipfian distributions in language. *Cognition*, 223:105038.

Guido M Linders and Max M Louwerse. 2022. Zipfs law revisited: Spoken dialog, linguistic units, parameters, and the principle of least effort. *Psychonomic Bulletin & Review*, pages 1–25.