Information Retrieval and Data Mining (COMP0084) Coursework 1

Student ID: XXXXXXXX

Abstract

Coursework 1 for Information Retrieval and Data Mining (COMP0084).

Task 1 – Text statistics

D1

Before the start of each tasks, some test preprocessing steps are applied including Tokenisation, Normalisation and Lemmatisation. After text preprocessing, the identified index of terms with size of **98563** can be built. Note that these steps are implemented by NLTK (Bird et al., 2009) library.

Tokenisation

The first step is to chop up a passage unit into pieces of "tokens," which enables further analysis and processing of text data.

Normalisation

In this step, the punctuation is removed since it is relatively meaningless in terms of NLP. Besides, all upper cases are changed into lower cases to reduce structural noise from corpus, e.g., "Apple" and "apple" won't be regarded as the same word if it doesn't be applied.

Lemmatisation

After applying tokens' normalisation, one way to improve the accuracy of search results for the retrieval model the following task is to revert words into their base forms. In the ways, one benefit is the reduction of size of vocabulary, which reduce the size from 128196 to 98563. The other merit is the decrease of noise in the corpus, e.g., "car" and "cars" are different tokens with same meaning, which can be viewed as same tokens after proper lemmatisation.

Text Statistic – Zipf's law

Zipf's law states that the frequency of a word is inversely proportional to its rank in the frequency table, which in the Zipfian distribution is defined by

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

where $f(\cdot)$ denotes the normalised frequency of a term, k denotes the term's frequency rank in our corpus (with $k = 1$ being the highest rank), N is the number of terms in our vocabulary, and s is a distribution parameter.

Setting $s = 1$ as requirements, the empirical distribution of the corpus and Zipfian distribution in the log-log plot are shown in Figure 1. We can observe that terms with high probability of occurrence tend to follow Zipf's law except for the few terms with low probability. For terms with occurrence probability lower than 10^{-5}, their frequency ranks are significantly lower than the theoretic value.

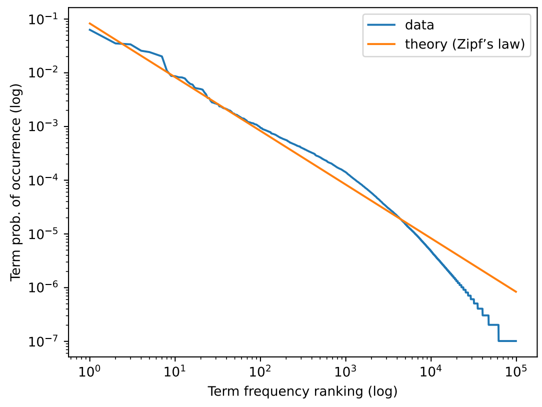

Figure 1: Empirical Distribution with Stop Words

After removing stop words (around 146 words) from the identified index of terms, it is obvious that most high ranking terms (rank > 10^2) are less probable to occur than the theory as shown in Figure 2.

After applying student's t-test(Student, 1908) to compare the difference between the empirical distribution and the theoretical distribution, we can observe that the statistic value increase from -267.8 to -185.7, which indicates the difference between them grow after removing stop words.

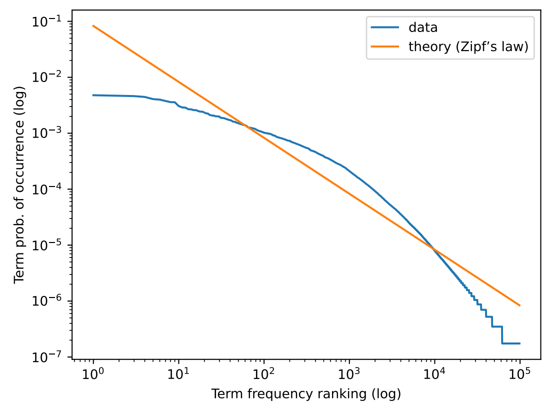

Figure 2: Empirical Distribution without Stop Words

Hence, as shown in figures, stop words tend to appear in most passages with high probability, which might decrease the accuracy of a retrieval model. Accordingly, **we will use the identified index of terms without stop words for the latter tasks**.

Task 2 — Inverted index

D3

The inverted index stores a mapping between each unique term and the passages in which it appears with **terms counts (frequency)**. In practice, I implemented this with nested hash map (2 layers), which the first layer use `term` as key, and the second layer use `pid` (document ID) as key with `terms counts` as values. The data structure is shown in Figure 3.

Besides, considering the fact that more than $180,000$ passages and more than $90,000$ terms will be the keys of inverted index, an auxiliary nested hash map is also stored in order to improve the retrieval speed and efficiency. In the auxiliary nested hash map, the first layer use `pid` (document ID) as key, and the second layer use `term` as key with `terms counts` as values. The data structure is shown in Figure 4.

In this way, the computing time can be greatly reduced. For example, using auxiliary nested hash map to count the document length should be faster than using inverted index since we don't have traverse the whole hash map.

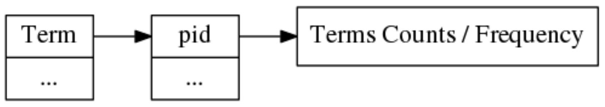

Figure 3: Inverted Index Hashmap

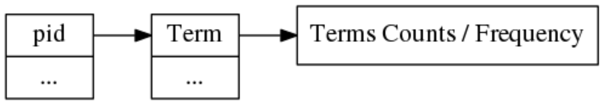

Figure 4: Auxiliary Inverted Index Hashmap

Task 3 – Retrieval models

Task 4 –- Query likelihood language models

D11

Question: Which language model do you expect to work better?

Overall, **Dirichlet smoothing** seems to be able to provide better results comparing with Laplace smoothing and Lidstone correction due to the consideration of background probability besides max likelihood estimate. For instance, in query "*what do you do when you have a nosebleed from having your nose*", both Laplace smoothing and Lidstone correction give the highest score to the passage with $pid = 2203469$, while the passage with $pid = 6178092$ is more relevant to the query, which is the passage with the highest score in Dirichlet smoothing.

Question: Which language models are expected to be more similar and why?

The similarity between models, Laplace smoothing should be similar to Lidstone correction because they both use max likelihood estimate to score passages. The only difference between both is constant weights to the unseen terms. For Dirichlet smoothing, it interpolates with background probability of a word so that the search results can be improved. In a simple experiment, if we only consider the passage with top score of each queries, Laplace smoothing and Lidstone correction choose the same passages in 61% of queries, while Laplace smoothing and Dirichlet smoothing select the same passages in just 38% of queries.

Question: Comment on the value of $\epsilon = 0.1$ in the Lidstone correction

Choosing $\epsilon = 0.1$ in the Lidstone correction seems to be a good choice since it alleviates the problem in Laplace smoothing, which give to much weight to unseen terms. From the simple experiment in previous paragraph, it shows that Lidstone correction and Dirichlet smoothing choose the same passages in 58% of queries.

Question: If we set $\mu = 5000$ *in Dirichlet smoothing, would this be a more appropriate value and why?*

According to the equation of Dirichlet smoothing, with larger μ, the smoothing effect will increase, which would not be an appropriate idea since the latter term (background probability) will dominate score value. It will cause Dirichlet smoothing model wrongly give more score to unrelated passages.

$$\frac{|D|}{|D|+\mu} * P(w|D) + \frac{\mu}{|D|+\mu} * P(w|C) \quad (2)$$

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

Student. 1908. The probable error of a mean. *Biometrika*, pages 1–25.