# COMP0084 Information Retrieval and Data Mining Coursework 1

**Anonymous ACL submission**

## Abstract

In this project, we completed 4 specific tasks aiming to explore methods for retrieving relevant passages from a collection given a query. In task 1, we preprocessed the passage collection and justified that the 1-gram terms follow Zipf's law. In task 2, an inverted index of the passages was constructed to be used for efficient retrieval of passages. In task 3, we extracted the TF-IDF representations of the passages, which were further used in the implementation of a simple vector space model and the BM25 model. Finally, several query likelihood models with different smoothing techniques (Laplace, Lidstone and Dirichlet) were implemented in task 4.

## 1 Introduction

### 1.1 Aim

Information retrieval modes leverage statistical techniques to rank and retrieve relevant documents from a large collection given a query. This paper aims explore and compare different retrieval models that solve the problem of passage retrieval by re-ranking candidate passages. More particularly, we aim to develop retrieval models that return the top 100 most relevant passages given a short query.

### 1.2 Data

Three data files are used in this project: passage-collection.txt, test-queries.tsv, candidate-passage-top1000.tsv. The passage-collection.txt file contains 182469 passages. The test-queries.tsv file is a tab separated file with each row containing a test query identifier (qid) and the corresponding query text. The candidate-passage-top1000.tsv gives the top 1000 (at most) candidate passages for each of the queries in the test-queries.tsv file.

## 2 Task 1 - Text Statistics

In this task, we first preprocessed the passage-collection.txt file, converting the raw text data into 1-gram terms. Then, we constructed a frequency table based on the processed data. Finally, we justified that the 1-gram terms follow Zipf's law by examining the relationship between the normalised frequencies and the frequency rankings of the terms.

| Term | Frequency | Term | Frequency |
|---|---|---|---|
| `the` | 618684 | `and` | 250988 |
| `of` | 332028 | `to` | 237481 |
| `a` | 275488 | `is` | 210086 |

Table 1: Top 6 most frequent terms in passages.

### 2.1 Text Preprocessing

The following preprocessing steps were carried out. First, all the passages were tokenized into 1-gram terms using the RegexpTokenizer. Then, all terms were normalized by converting lowercasing and removing terms with non-alphabetic characters (e.g. numbers, punctuations, etc.). Normalization ensures that capitalization does not affect the model performance and also reduces noise in the data. The final step was to remove the stop words when required.

### 2.2 Frequency Table

After preprocessing the text data, we obtained 182469 terms and the size of the identified index of terms is 88291. We then implemented a function that counts the occurrences of these terms and constructs a frequency table. Table 1 shows the top 6 most frequent terms identified in the passages.

### 2.3 Frequency Distribution

In figure 1, we demonstrates the inverse proportional relationship observed between the normalized frequency and the frequency ranking for the top 1000 most frequent terms in the passages. This roughly suggests that these terms follow Zipf's law. Only the top 1000 ranking terms were plotted in

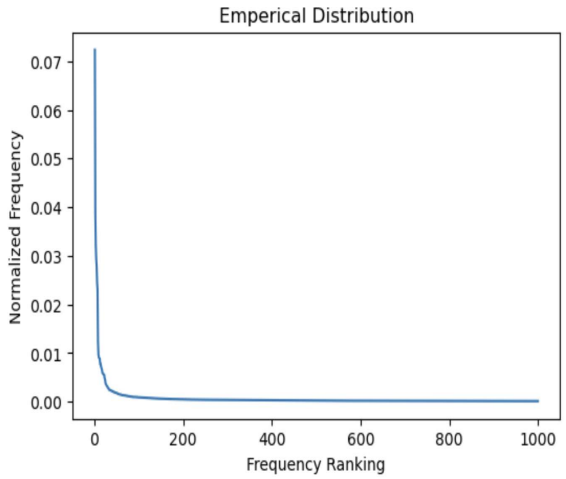


Figure 1: Normalized frequency against frequency ranking of the top 1000 ranking terms in the passages.

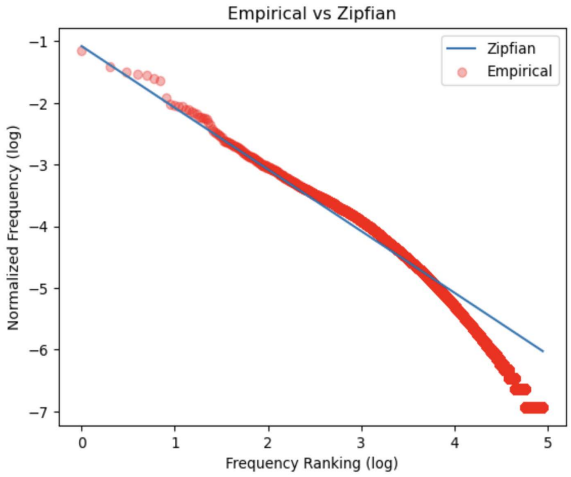


Figure 2: Empirical distribution of the terms and the actual Zipf's distribution.

order to obtain a better zoomed diagram. To further examine the difference between the empirical distribution observed and the actual Zipf's law distribution, we visualized the two distribution in a log-log plot as shown in figure 2. At the high frequency end, the top 7 ranking terms do not follow the Zipf's law strictly. More specifically, most of them have frequencies higher than expected. At the low frequency end, there exist a large number of terms with frequencies much lower than expected. This might be due to the small vocabulary size of the passages which contradicts with Zipf's law's assumption of an infinitely large corpus.

We extended our examination on the frequency distribution of terms by removing the stop words. Stop words are the most frequently occurring terms in most text data, therefore, removing them resulted

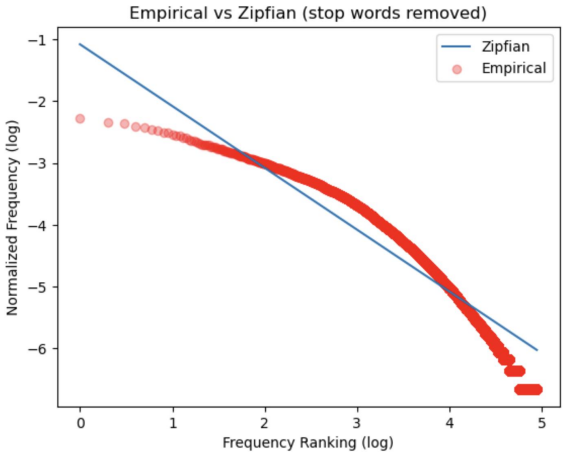


Figure 3: Empirical distribution of the terms (with stop words removed) and the actual Zipf's distribution.

in a downward bent curve at the top end, as shown in figure 3. Moreover, the normalized frequencies of some mid-ranked terms became higher than the Zipf's law suggested. This is because the total number of terms was reduced largely when the high frequency stop words were all removed.

## 3 Task 2 - Inverted Index

In this task, we built an inverted index for the passages storing both the frequency and positional information of each term. We first implemented two step to further preprocess the passages: removing stop words and lemmatization. Lemmatization reduces the terms to their contextualized base forms, which always produces valid words and is therefore more accurate than stemming.

We use a dictionary to construct the inverted index with the keys being the terms and the values being the corresponding information dictionaries. Within an information dictionary for a term, the keys are the identifiers (pids) of the passages that contains the term and the values are lists containing two elements: frequency, position.

The inverted index was built through the following steps:

1. Map each preprocessed passage to its corresponding identifier ("pid").

2. Loop through each term in the passage, count the number of occurrence and record the position.

3. Update the inverted index dictionary with the key being the term and the value being an information dictionary: "pid": [frequency, position].

4. Loop through all the passages and continuously update the inverted index dictionary.

This inverted index is further used in later task for efficient retrieval of passages.

## 4 Task 3 - Retrieval Models

In this task, we first extracted the TF-IDF vector representations of the passages and the queries. Then, we constructed a simple vector space model and a BM25 model to retrieve the top 100 relevant passages for each query.

## 5 Task 4 - Query Likelihood Language Models

In this task, we constructed 3 query likelihood language models with different smoothing techniques: Laplace smoothing, Lidstone correction and Dirichlet smoothing. We then used these models to retrieve the top 100 relevant passages for each query.

Query likelihood language models (QLLM) are a family of probabilistic models used in information retrieval to estimate the the probability of observing a document given a query. To prevent obtaining a probability of zero, QLLMs use smoothing techniques to address the problem of sparsity where some terms in the query may or document may not exist in the corpus.

### 5.1 Smoothing Techniques

Laplace smoothing is a simple and straightforward smoothing technique. It assumes that every term is equally likely to appear in any document. Therefore, Laplace smoothing adds a fixed value (usually 1) to the count of every term to account for unseen terms. Its simple nature empowers it to be fast in computation but may overestimate the probability of rare terms.

Lidstone correction is a generalization of Laplace smoothing that uses a smoothing parameter between 0 and 1 instead of a fixed value to control the degree of smoothing. Generally, Lidstone correction is more effective than Laplace smoothing but the actual performance depends largely on the specific dataset and the choice of parameter.

Both Laplace smoothing and Lidstone correction can avoid zero probability to occur, however, they are unable to make valid estimations for unseen terms. Dirichlet smoothing is an improved method that solves this problem by taking into account the background probability of a term, which is the probability distribution of the term in the collection of documents.

### 5.2 Model Performance

Laplace smoothing is the simplest method among the three which assumes that every term is equally likely to appear in any document. This clearly does not hold as we had demonstrated the terms follow Zipf's law in task 1. Therefore, it is likely to perform the worst.

In the rest two methods, we expect Dirichlet smoothing to outperform Lidstone correction because the former is a more advanced method that adjusts its computation of probability dynamically according to the input term. From the retrieval results, we observed that for the query with $qid = 1108939$, only Dirichlet smoothing retrieved a common passage $pid = 2846462$ which was ranked the first in both the consine similarity and BM25 models, within the top 5 retrieved passages.

### 5.3 Similarities

As mentioned above, Laplace smoothing is expected to perform the worst and therefore we expect Lidstone correction and Dirichlet smoothing to be more similar. This is because that the selections of parameter values of these two methods are likely to result in a similar degree of smoothing while Laplace smoothing will obviously result in over-smoothing.

For example, within the top 5 most relevant passages retrieved for the query with $qid = 1108939$, models with Lidstone correction and Dirichlet smoothing both retrieved the passages with $pid = 6707713, 8596285, 3130232$, while the model with Laplace smoothing did not retrieved any common passage.

### 5.4 Model Parameters

Setting $\epsilon = 0.1$ for Lidstone correction is a suitable choice. Larger values of $\epsilon$ lead to more smoothing, while smaller values lead to less smoothing. The parameter $\epsilon$ in Lidstone correction should vary between 0 and 1. While a small value like 0.1 will lead to less smoothing, it prevents the model from being heavily affected by the assumption of uniform distribution as that in Laplace smoothing.

Setting $\mu = 5000$ would not be a good choice when implementing Dirichlet smoothing. Generally, longer passages will require less smoothing than short ones. Therefore, the parameter in Dirichlet smoothing is often set to a similar value to the average document length. In this case, we have an average document length of around 25. Thus,

$\mu = 50$ would be a more appropriate value.

However, despite the above statements, it is always a good practice to select the most appropriate parameter through tuning the model on a validation dataset rather than picking values based solely on the theory.