# COMP0084 Coursework 1 report

**Anonymous ACL submission**

**Abstract**

Information Retrieval method are wildly using many field such as search engine data mining, this paper is focusing on giving each query a ranked passage that best bit the query by using retrieval model method.

## 1 Introduction

This paper shows an experimental study of implementing a information retrieval model to solve passage retrieval depend on passage-query pair from file "candidate-passages-top1000.tsv" and query file "test-queries.tsv", this papaer discussed the implementation of solving the retrieval problem step by step, includes: data preprocessing, creating inverted-index, retrieval with TF-IDF model, retrieval with Query likelihood language model.

## 2 Tasks

### 2.1 Task1: Text statistics

#### 2.1.1 D1

My choice of text preprocessing contains follow steps:

1. `Tokenisation`:
   splitting terms within the document into list of tokens. And remove punctuation symbols within tokens, symbols are being heavily duplicate among all passages.

2. `Normalisation`:
   My choice is to discard meaningless white spaces terms and transfer all tokens into lower case.

3. `Stemming`:
   In this experiment perform stemming before perform stop word removal, stemming is to reduce tokens of derived tokens into its root form. The reason is after stemming some of the words could be transformed into stop words, and need to be removed after since it represents no meaning for the passage.

4. `Stop word removal(optional)`:
   In the case of tokens are all uni-gram, tokens of "this","and","is" is meaningless, so are considered stop words, tokens that are stop word are removed from the vocabulary.

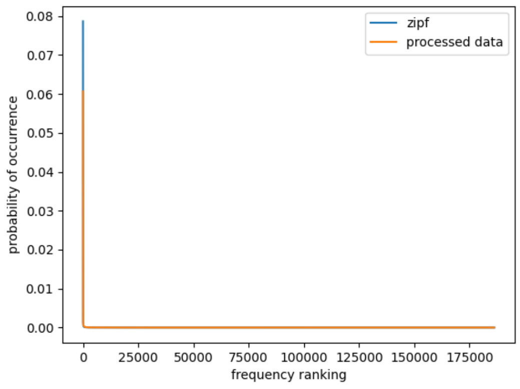

Figure 1: number of occurrences against frequency ranking

There is also a technique of Lemmatisation, it returns all words to its base form but need vocabulary and morphological analysis, so this experiment will not use this approach. After applying preprocessing, the vocabulary size without stop words removal is 186191, with stop words removal is 186061.

Apply zipf's law with s = 1:

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

After text preprocessing, plot the number of occurrences of tokens against their frequency ranking with zipf's law distribution: See Figure 1, its difficult to compare two lines but shows that the occurrences against ranking are very similar to zipf

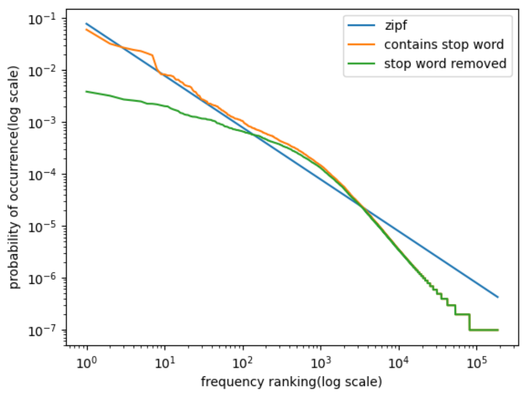

Figure 2: number of occurrences against frequency ranking in log scale

distribution. Then plot the figure is log scale with new line represent the plot with stop word removal in Figure 2. We can see clearly that the passages without stop word removal are really close to sipf distribution, but for tokens of low frequency ranking are much lower than zipf distribution, the reason cause the difference is that the passage are not all identical sentences, for the one property the passage can explain it in different synonyms, e.g. car, automobile, motor, truck. Take the considering that car is a highly use words and should be classified into high frequency ranking token, but when represent car with it's synonyms, its probability of occurrence decrease and its synonyms can be classified as low frequency ranking words, result probability of low frequency ranking probabiliry slightly lower that zipf law. Inconsistency in the expression of the same thing caused the bias. After applied stop word removal, probability of token with higher ranking slightly decrease, it shows that stop word are very frequently exist in passages, but with no actual meanings, hence remove them should improve performance of the model.

## 2.2 Task2

### 2.2.1 D3

An inverted index provide an inverted tracking method which can reduce calculation cost of models. In the following models, there are many cases the model are counting the appearance frequency of a token so call token frequency(tf). Inverted index provids the functionality of a dataset to search tf of a token within a specific passage. The algorithm is implemented by dict counter structure and use token and qid to track the tf of the token appears in a passage, the algorithm output data structure shows as follow: {token:{passage id: tf with this passage}}

$$\{"definit" : \{"7130104" : 3, "8002085" : 1...\}\}$$

$$\{"rna" : \{"7130104" : 5, "8466138" : 1...\}\}$$

$$\{"along" : \{"7130104" : 2, "7133139" : 1\}\}$$

The inverted index provided a method to perform fast full text search, models in Task3 runs for 30-40 minutes without applying inverted index. Inverted index decreased the time spend into less than 2 minutes.

## 2.3 Task3

### 2.3.1 D5-6

bm25.csv and tfidf.csv provided

## 2.4 Task4

### 2.4.1 D8-10

file laplace.csv, ladstone.csv and dirichlet.csv provided

### 2.4.2 D11

Given that algorithm of three models are:
**laplace smoothing:**

$$\frac{m_1+1}{|D|+|V|}, \frac{m_2+1}{|D|+|V|} \cdots \frac{m_{|V|}+1}{|D|+|V|}$$

where D is a passage, |D| is the length of the passage,|V| is the length of vocabulary, laplace smoothing gives weight to all terms include unseen terms.

**lidstone correction:**

$$P(w|D) = \frac{tf_{w,D}+\varepsilon}{|D|+\varepsilon|V|}$$

where D is a passage, $tf_{w,D}$ is the token frequency(occurrence frequency of a token in a passage). instead give 1 to all terms, lidstone correction gives really small weight to all terms:$\varepsilon$ is the weight that given to unseen terms and reduce weight of the rest.

**Dirichlet Smoothing:**

$$P(w|M_D) = \lambda \cdot P(w|D) + (1-\lambda) \cdot P(w|C)$$

$$\lambda = \frac{N}{N+\mu}$$

$$P(w|M_D) = \frac{|D|}{|D|+\mu} \cdot P(w|D) + \frac{\mu}{|D|+\mu} \cdot P(w|C)$$

where $P(w|M_D)$ is the probability of the vocabulary model generate token w, $P(w|C)$ is the probability of the vocabulary(considering tf) generate w, $P(w|D)$ is the probability of generating w from current passage, in this N is the length of the document D.

**Answers:**

In this case, Dirichlet Smoothing is expected to perform better than other two models, the reason is: discounting methods laplace smoothing and lidstone correction are treating all unseen words equally, which means that all unseen words gains an extra weight equally, considered with Figure 1 words are having different occurrence frequency, which means that equally assign weight to unseen words is not a good strategy, but Dirichlet Smoothing method is considering the occurrence probability of unseen words, the probability unseen word fit zipf's law, hence Dirichlet smoothing method performs better.

Both as discounting method, lidstone correction and laplace smoothing performed similarly, this can be proved by there output [qid,pid,score] files are having really close scores, and they are both adding weight to unseen words.

In this case, compare with the document length, the average passage length is 32.97 after preprocessing, but the vocabulary is really big (186061), which means the $\varepsilon$ should be very small, i think choose $\varepsilon = 0.1$ is reasonable but could be more smaller.

According to Dirichlet Smoothing's equation, large $\mu$ will reduce $\lambda$ and the passage avg passage length is 32.97, $P(w|M_D)$ is:

$$\frac{32.97}{32.97+5000} \cdot P(w|D) + \frac{5000}{5000+32.97} \cdot P(w|C)$$

$$= 0.0065P(w|D) + 0.9935P(w|C)$$

which means that the model will generate word w only depend on the global vocabulary tf regardless of the corresponding passage. Hence select $\mu = 5000$ is not a good choice.