Information Retrieval and Data Mining (COMP0084) Coursework 1 Report: Passage Ranking System

Anonymous ACL submission

1 Text statistics

In this section, the preprocessing process is introduced and the analysis of the text statistics results of the observed distribution and the Zipf's distribution is reported.

1.1 Preprocessing process

Initial inspection of the passage dataset finds that 'Best Answers:' exists in some of the passages and the content after the colon (:) is the useful response message, thus the preprocessing process only stores the message after the colon if 'Best Answers:' is found to keep the response message clean.

Secondly, the preprocessing process removes punctuation and the emoji pattern such as 'dog face' from the passage to reduce the size of redundant vocabulary. Since most of the punctuation and emoji pattern are connected with the words without any space, for the same word 'floor', 'floor.' (with punctuation) and 'floor\u1F415' (with emoji pattern) are identified as different words.

In addition, the preprocessing process replaces multiple continuous spaces, using regex pattern (\s\s+), with single space to prevent the python `split()` method from producing empty strings. The preprocessing process further removes special Unicode character such as '\u200b', representing 'zero width space' (U+200B) from the passage to eliminate redundancy in the vocabulary.

After the preprocessing process, the identified vocabulary size is 143806.

1.2 Report of statistics result

The plot of the probability of occurrence (normalised frequency) against the frequency ranking for this corpus of passages is shown as in figure 1. The word frequency can be observed as a hevily-tailed distribution, which can be modelled by Zipf's law with $s = 1$. The shape of the empirical distribution is compared against the shape of the Zipf's distribution in figure 2 which the two shape overlaps to further prove that the terms follow Zipf's law.

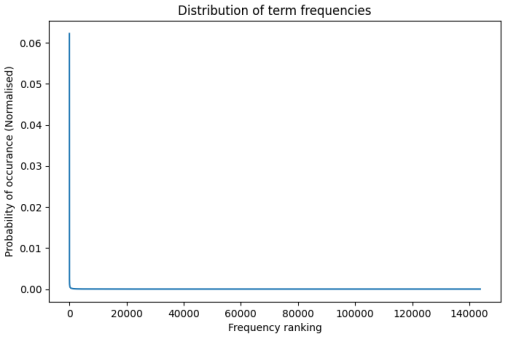

Figure 1: Plot of the probability of occurrence (normalised frequency) against the frequency ranking

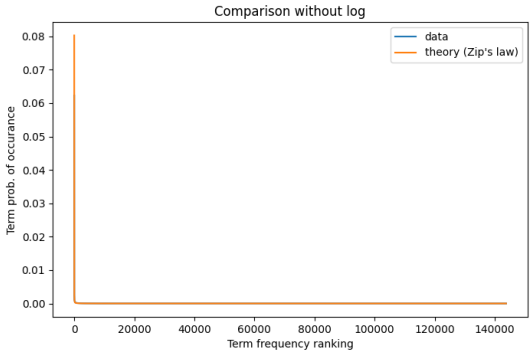

Figure 2: Comparison of the empirical distribution and the Zipf's distribution without log-log plot

The comparison between the empirical distribution and the actual Zipf's law distribution is shown in figure 3. The Zipf's law for text sets $s = 1$ in the Zipf distribution is defined by:

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

Using the eqation 1, we can observed that when $s = 1$, the log of the Zipf's law distribution transforms into a straight line with a decreasing trend.

While the log of the empirical distribution fluctuates in in different parts, especially the tailed part. This is due to the fact that Zipf's law is an ideal distribution that the term frequency is proportional to the inverse of its ranking. For real-world corpus, this assumption is rarely true. As we can observe in the figure 3, the difference is gradually growing as the term frequency ranking increase. At the tailed part where the empirical distribution drops vastly, the difference reaches its maximum since the real-world corpus contains a lot of uncommon words with the same term frequency 1, which is a constant and cannot be proportional to the increasing ranking anymore.

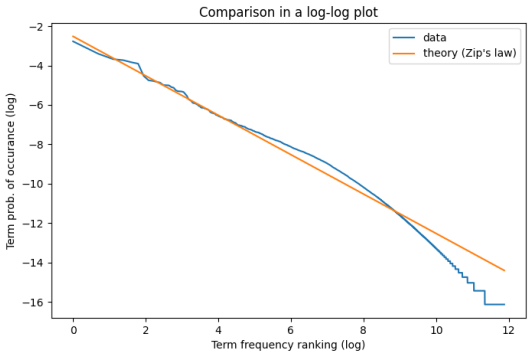

Figure 3: Comparison between the empirical distribution and the actual Zipf's law distribution in a log-log plot

2 Inverted index

In this section, the generation of the inverted index is introduced with the justification of the choice.

To generate an inverted index for this task, the *document-level* index is chosen. Before the generation of the inverted index, the aforementioned preprocessing process in Task 1 is applied, followed by the removal of the stop words. The list of stop words used for this task is downloaded from the NLTK corpus of stop words. Besides, the term occurrence calculated in Task 1 for each term in the vocabulary is loaded as a term-frequency dictionary.

After finishing the preparation, the generator of the *document-level* inverted index records, for each term, all the **pid** (unique passage identifier) that term appears with its corresponding number of occurrence in each passage. The entire inverted index is stored using the structure of 'a dict of dict', which the first level of keys is the vocabulary and the second level of keys are all **pid** with the appearance of the term, pointing to its number of occurrence. When the generation of the inverted index is done, it gets stored to a local pickle file named `inverted_index.pkl` for later use in Task 3 and 4. A slice of the inverted index is shown as below:

```
"barley": {
    2584239: 4,
    4733434: 2
    }
```

The advantage of an *document-level* index is that it is much smaller in size than the *word-level* index storing the position of each term. Also, the cost of storing the common words using *document-level* index is dramatically smaller than using the *word-level* index as mentioned in the following quotation.

> An aspect of inverted indexes that is dramatically altered by the introduction of word positions is the cost of processing common words. In a *document-level* index, common words such as *the* are relatively cheap to store. In a *word-level* index, the average per-document requirement for common words is much larger because of the comparatively large number of $f_{d,t}$ word-gap codes that must be stored. (Zobel and Moffat, 2006)

Rather than knowing the information about the precise location of the appearance of a term in a passage, the information of the number of occurrence is sufficient for the following task to calculate TF-IDF value, BM25 score and build a query likelihood language model. However, only recording the information about the **pid** that each term occurs in is not enough for calculating the BM25 score in Task 3 or building a query likelihood language model in Task 4. In conclusion, the choice of using a *document-level* inverted index is made for this task.

3 Retrieval models

In this section, the description of different retrieval models implemented in Task 3 using TF-IDF vector representation and BM25 are given, respectively. Some of the example query and passages pairs are shown in Appendix A.

3.1 TF-IDF

The TF-IDF vector representations of the passages should be generated for this deliverable.

In order to utilise the computational resources, the TF-IDF vector is specifically generated for each query to limit the dimension of the vector to ($NumberOfPossiblePassages$, $VocabSize$), usually $NumberOfPossiblePassages = 1000$ and the $VocabSize$ for this task is 143663.

Computing the TF-IDF value is composed of two parts: TF (Term Frequency) and IDF (Inversed Document Frequency). Knowing TF is straightforward since $TF_{(t,d)}$ = Number of times the term occurs in a document, which is already computed and saved in Task 1. Computing IDF requires the information about the number of documents that the term occurs, which the formula of IDF is:

$$IDF_t = \log(\frac{\text{Total number of the documents}}{\text{Number of documents with the term}})$$

The inverted index saved in Task 2 gives the information about the number of documents with the term, thus we can use the inverted index directly to compute IDF value. After both values are calculated, TF-IDF value for each term in the passage can be calcuated using TF $\times$ IDF. The same process of the calculation of the TF-IDF values happens for the query.

The vector respresentations of the passages are then built by expanding the TF-IDF value computed for each term to a one-dimensional vector of the $VocabSize$ and concatenating all the one-dimensional vector to construct the final two-dimentional vector of size ($NumberOfPossiblePassages$, $VocabSize$). The vector of query is built using the TF-IDF values and expanding them into a one-dimensional vector of the $VocabSize$ to allow the calculation of the consine similarity.

For each possible passage, the consine similarity is calculated between the query and the passage and the resultant score is stored. By sorting the scores for all passages in a descending order, the top 100 passages can be retrieved easily.

3.2 BM25

Differently to the TF-IDF value, the calculation of BM25 score makes use of the number of occurrence of the term in a document. The score of a document D given by a query Q of length n is given by:

$$\begin{aligned}&\text{BM25}(D,Q) \qquad (2)\\ &= \sum_{i=1}^{n} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)}\\ &\cdot \frac{(k_1+1)f_i}{K + f_i} \cdot \frac{(k_2+1)qf_i}{K + qf_i}\end{aligned}$$

where $K = k1 \cdot ((1-b) + b * (\frac{dl}{avdl}))$ which dl is the document length and $avdl$ is average document length. N is the total document size. f_i and qf_i are the number of occurrences of the term in the document and the query respectively. R and r are the relevence information parameters which are 0 in this task. $k1$, $k2$ and b are free parameters that are given in the task.

For each query Q, the BM25 gives a dictionary of scores for each passage in the possible passages list, with **pid** as the keys and the scores as the values. Then the dictionary is sorted in a descending order to allow the retrieval of the top 100 passages.

4 Query likelihood language models

In this section, the description of each query likelihood language model with different smoothing method are given, followed by the discussion of the smoothing techniques and their choice of hyperparameters.

4.1 Models

Laplace smoothing Laplace smoothing is the simplest smoothing method used in this task. The smoothing method simply adds 1 to each number of occurence m_i and rather than dividing by the document length $|D|$, it divided by $|D|+|V|$ where $|V|$ is the vocabulary size. The Laplace smoothing estimates:

$$\text{Laplace} = \prod_{1}^{n} \frac{m_i + 1}{|D|+|V|} \qquad (3)$$

For each passage in the possible passage list of each query, the passage receives a log-based Laplace-smoothed score and then the scores are sorted in a descending order to allow the top 100 passages to be given.

Lidstone correction Since Laplace smoothing is giving too much weight to unseen words, Lidstone correction improves on that and gives a control to the vocabulary size. The Lidstone correction examines for a query of length n:

$$\text{Lidstone} = \prod_{1}^{n} \frac{TF_{(w,D)} + \epsilon}{|D|+\epsilon|V|} \qquad (4)$$

where $TF_{(w,D)}$ is the term frequency of each word in the document and $\epsilon = 0.1$ is the hyper-parameter.

For each passage in the possible passage list of each query, the passage receives a log-based Lidstone-corrected score and then the scores are sorted in a descending order to allow the top 100 passages to be given.

Dirichlet smoothing Dirichlet smoothing is an interpolation methods that improves Jelinek-Mercer smoothing. For the Dirichlet smoothing, it estimates not only the probability of the seen words but also the unseen words as a background probability. The probability of the seen words and the background probability are controlled by a parameter λ, which in the dirichlet smoothing, λ is calculated depending on the document length $|D|$. The dirichlet smoothing estimates for a query of length n:

$$\text{Dirichlet} = \sum_{1}^{n} \log\left((|D|/(|D|+\mu))\frac{TF_{(w,D)}}{|D|} + (\mu/(|D|+\mu))\frac{cf_{(w)}}{|V|}\right) \tag{5}$$

where $cf_{(w)}$ is the term frequency of the term in the corpus and $|V|$ is the vocabulary size.

For each passage in the possible passage list of each query, the passage receives a Dirichlet-smoothed score and then the scores are sorted in a descending order to allow the top 100 passages to be given.

4.2 Discussion

Dirichlet smoothing is the one expected to work better than Laplace smoothing and Lidstone correction since the other two are discounting methods. Discounting methods have the nature of treating each unseen words equally (adding 1 or ϵ), although some words are obviously more frequent than others. Also, Dirichlet smoothing includes more

The results of Laplace smoothing and Lidstone correction are expected to be similar since they both are discounting methods. The scoring formula of the two methods are similar, in fact, if the ϵ in Lidstone correction is set to 1, it becomes Laplace smoothing.

5 Cached additional files

In order to avoid duplicate calculation, some of the important results are cached to local files during the runtime to be used in later tasks. The list of files are as follows:

`term_dict.json`: The dictionary stores the term with its corresponding term frequency, i.e. number of occurrence, in the corpus. The file is generated and cached in `task1.py`.

`inverted_index.pkl`: The inverted index of the terms in the vocabulary. The file is generated and cached in `task2.py`.

Besides, some images are stored as local PNG files in `task1.py` for the plots required in Task 1. The list of images are as follows:

`Distribution of term frequencies.png`: The plot of the probability of occurrence (normalised frequency) against the frequency ranking of the vocabulary.

`Comparison in a log-log plot.png`: Comparison of the empirical distribution against the actual Zipf's law distribution in a log-log plot.

`Comparison without log.png`: Comparison of the empirical distribution against the actual Zipf's law distribution without the log-log plot.

References

U+200B. unicode character 'zero width space' (u+200b).

Justin Zobel and Alistair Moffat. 2006. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2):6–es.

A Task 3 Example Result Tables

Query	Passage	Score
what slows down the flow of blood	"Hypo-function of the thyroid can cause everything we just talked about because: 1 it slows the rate of glucose uptake by cells; 2 it decreases rate of glucose absorption in the gut; 3 it slows response of insulin to elevated blood sugar; and, it slows the clearance of insulin from the blood." (pid: 4390191)	0.461075
what slows down the flow of blood	"Blood flow, blood pressure, resistance. - blood flow: volume of blood flowing through vessel/organ/ circulation per minute; as. far as systemic circulation, blood flow = CO. - blood pressure: pressure gradient between 2 points in vasculature. -resistance: opposition to flow due to friction." (pid: 2387200)	0.453905
what slows down the flow of blood	"1 This means that more blood rubs against the walls of the vessel and it slows blood flow. 2 In any one capillary, this resistance is an advantage because the slowed blood flow has more time for gas exchange to occur. 3 When an arteriole dilates, the diameter almost doubles." (pid: 2846462)	0.453598

Table 1: Passage Ranking System using TF-IDF vector representation. In the table the top 3 passages is shown for qid = 1108939

Query	Passage	Score
what slows down the flow of blood	"Esophageal varices are swollen blood vessels in the esophagus (swallowing tube), the tube that connect the mouth to the stomach ... Esophageal varices occur when normal blood flow to your liver is slowed. Liver disease may create scar tissue in the liver which slows the flow of blood ..." (pid: 2068541)	27.886290
what slows down the flow of blood	"Blood flow often slows in the bulging section of an aortic aneurysm, causing clots to form. If a blood clot breaks off from an aortic aneurysm in the chest area, it can travel to the brain and cause a stroke. Blood clots that break off from an aortic aneurysm in the belly area can block blood flow to the belly or legs." (pid: 3130232)	27.803400
what slows down the flow of blood	"Idiopathic pulmonary fibrosis (IPF) causes scar tissue to grow inside your lungs ... IPF scar tissue is thick, like the scars you get on your skin after a cut.It slows oxygen flow from your lungs to your blood, which can keep your body from working as it should." (pid: 6707713)	27.508764

Table 2: Passage Ranking System using BM25. In the table the top 3 passages is shown for qid = 1108939