Information Retrieval and Data Mining Coursework 1

20042465
University College London
London, UK

ABSTRACT

With the advance of technology and the emergence of big data, the world can generate millions of information a day. How to find the "key information" from these tens of thousands of information is always a very significant topic. Information retrieval plays an essential role in this field. The information retrieval system is a facility between information users and information collection. For an information problem, the function of the information retrieval system is to retrieve the desired and filter out the unwanted data, and further to reduce the information overload.

In this coursework, we will calculate the correlation between the query (from the users) and its corresponding passages, compare the different information retrieval models, and return the higher relevant passages in descending order. During the procedure, in the first place, we will analyze some of the recent developments in passage-ranking models and further check whether it follows the Zipf's law for a collection. Moreover, we will implement the inverted index and ranks candidate passages for each query with the Vector Space Model, BM25 and Language Modelling (LM-Dirichlet, LM-Laplace, LM-Lidstone) in the end.

1 TEXT STATISTICS

Text pre-processing is traditionally a critical step for cleaning and preparing text data for use in information retrieval [2]. After the pre-processing, we can reduce the text to only the words that we need in the collection.

Therefore, to make the model more efficient, in this section, we use the **nltk.word_tokenize** [9] to split the words and do the following pre-processing steps (the detailed reasons will be explained in **Inverted Index** section):

- Convert all of the words into the type of lowercase.
- Remove the blank.
- Use the **string.punctuation** to remove the punctuations.
- Use the **re.search** to remove the digits [11].
- Use the **PorterStemmer** to remove the suffix of the words (normalization) [7]. For instance: the word "connection" and "connected" will become the word "connect".

In addition, we combine all passages into single list, record the key of each terms and calculate the frequency of each terms during the procedure.

1.1 Zipf's law

Zipf's law is a law about the frequency distribution of words in a collection with these parameters, the rank r_t, the frequency f_t and the constant K. The rank r_t depends on the frequency f_t, when the word possesses larger frequency will have the smaller rank value, vice versa. They have the inversely relationship [15].

Hence, the Zipf's law can be stated as:

$$r_t * f_t = K \tag{1}$$

On top of that, we also can express the formula as below:

$$r_t * Pr_t = \frac{K}{N} = C \tag{2}$$

Where Pr_t is the probability of the word occurrence, and N is the total number of the words in the collection (not number of unique words) and the C is the constant.

After the calculation, we can create the statistical table which includes every word information, in addition, we do not consider the word whose frequency is smaller than 3 times in our table, because, we suppose that the word whose frequency is smaller than 3 times maybe is the incorrect spelling word or the error word, which would have an impact on the statistical table distribution.

In this section, we use the **pd.Dataframe** [5] to exhibit the top 15 words (most frequency) of the statistical table as below:

	Word	Frequency	Rank	Probability(%)	Rank * Probability
0	the	622419	1	6.132931	0.061329
1	of	333419	2	3.285304	0.065706
2	a	279341	3	2.752453	0.082574
3	and	253633	4	2.499142	0.099966
4	to	239433	5	2.359224	0.117961
5	is	217327	6	2.141405	0.128484
6	in	198978	7	1.960606	0.137242
7	for	107559	8	1.059820	0.084786
8	you	86004	9	0.847430	0.076269
9	or	85216	10	0.839666	0.083967
10	it	81385	11	0.801917	0.088211
11	that	80927	12	0.797404	0.095689
12	are	77398	13	0.762632	0.099142
13	as	67491	14	0.665014	0.093102
14	on	67394	15	0.664059	0.099609

Figure 1: top 15 - most frequency words (distribution)

On the basis of the above statistical table, we can compute the sum of the $r_t * Pr_t$ and divide the N, the total number of the words in the collection (mean value) to obtain the constant C. Finally, we can obtain the value of $C = 0.083 \pm 0.03$.

$$\sum_{t=1}^{N} \frac{r_t * Pr_t}{N} = C \tag{3}$$

In fact, the Zipf's law in the ideal situation, the value C can equal to the value 0.1, and the slope of the $log(Pr_t) - log(r_t)$ plot is equal to -1, we will explain the reasons as follows [3].

Firstly, we can rewrite the equation (2) into a power law where a = 1, as below:

$$Pr_t = \frac{C}{r_t^a} \tag{4}$$

Next, we take the logarithmic value from both sides:

$$\begin{aligned} \log Pr_t &= \log(\frac{C}{r_t^a}) \\ &= \log(C) - \log(r_t^a) \\ &= \log(C) - a * \log(r_t) \end{aligned} \tag{5}$$

In the end, through the mathematical conception, the line of the equation can be expressed as $y = mx + b$, as a result, we can understand that the the $log(Pr_t) - log(r_t)$ plot is the line with the slope $-a = -1$.

Therefore, we can use the equation (2) with the slope = -1, and the constant $C = 0.1$ to write the ideal $log(Pr_t) - log(r_t)$ plot of the Zipf's laws and further to compare with the real $log(Pr_t) - log(r_t)$ plot as below:

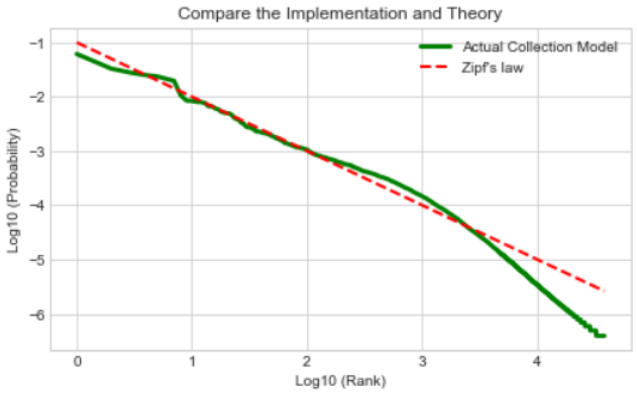

Figure 2: Compare with the ideal and the Actual Collection Model under the logarithmic plot.

According to the above plot results and computation results, we can realize that the Actual Collection Model approximately conform to the Zipf's law. Whereas, we also can notice that the plot exist some errors at the head and the tail. Because, in the light of the Zipf's law equation (1), we can know that the Zipf's law is more suitable for the word which possesses the intermediate frequency of the collection in most situation, hence, we can find that the curve are almost coincide with each other at the $x-axis\ log$ value 1 to 3 and will appear the higher deviation value at the larger and the lower frequency.

2 INVERTED INDEX

Inverted index is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines [12]. The main method to implement the inverted index is to take the word as the key index, and further to record each word appears in which documents (passages) and its frequency. What's more, we also can record any relevant information during this procedure. The inverted index contains numerous advantages, some reasons illustrating my perspectives will be shed light on as follows:

(1) The objective of the inverted index is to pre-compute many values as soon as possible and reduce the complexity in the full-text searches.
(2) They can do so at a cost of increasing processing whenever a document goes on the database.
(3) It is uncomplicated to manage and develop.

Before we construct our inverted index system, we also need to implement the text pre-processing step, in this step, we add the extra pre-processing manners compare with the **Question 1**:

- Remove the url link.
- Convert all of the words into the type of lowercase.
- Remove the blank.
- Use the **string.punctuation** to remove the punctuations.
- Use the **re.search** to remove the digits [11].
- Use the **nltk.corpu.stopwords** to remove the stop words [10] which are meaningless and do not offer much information, such as "is", "a", "the", etc.
- Use the **PorterStemmer** to remove the suffix of the words (normalization) [7]. For instance: the word "connection" and "connected" will become the word "connect".

The reason why we remove the url links, thanks to in the text, the url links are reference to a location on the web, but do not offer any additional information. Furthermore, we remove the digits and the punctuation which do not add any value, due to they are pointless. Next, we convert all the words into the lowercase, hence, we can classify the same words (different formats) into the same category. We also remove the stop words, because of stop words such as "is", "a", "the" are words which appear with a very high frequency in a passages and do not furnish useful information for the context or the true meaning of a sentence. Since there are many variations of words that do not bring any new information and create redundancy, the last step we use the stemming technique to remove the suffix of the words (normalization). These text pre-processing steps not only can reduce the vocabulary size but also can reduce the complexity to improve the efficiency.

After the text pre-processing, we utilize the python *dictionary* data structure to create the inverted index, we calculate all the passages (pid) which are related to each query (qid), in other words, the each query only need to focus on their corresponding passages. Additionally, the each term will be the key index, and we store the pertinent information in our class *Query _ Result* with dictionary structure and show its information as follows:

(1) Record the each term occur in which passages (pid).
(2) Record the each term frequency at each passages (pid) .
(3) Record the length of each passages (pid).
(4) Record the count of whole passages in the collection.

(5) Store the passages count of each query (qid).
(6) Calculate sum of the (1) information, which means that the sum of the each term occurs in the whole related passages.

The reasons why we store the information as above owing to we can compute the TF-IDF value more conveniently and make us to execute on the following questions more easily. We will explain the TF-IDF conceptions more detailed in the following **Question 3**.

3 RETRIEVAL MODELS

The final goal of the retrieval model is to choose and rank the relevant passages with respect to the user's query. The texts of the passages and the queries are represented in the same way, so that we can formalize the passage selection and ranking through the matching function and further to return the retrieval status value for each passage in the collection [1]. Accordingly, in this section, we will employ the previous inverted index data structure (**Question 2**), analyze and cooperate with the two common retrieval models – **Vector Space Model** and **BM25 Model**, and list the top 100 passages at each query at last.

3.1 TF-IDF

TF-IDF is a statistical method that generally applied in the information retrieval and the text exploration. It is used to evaluate the importance of the terms to passages. Hence, the crucial parts of the TF-IDF method are terms and passages. Moreover, we can compute the similarity between the two passages more simply. The TF-IDF method contains the following two components [4]:

(1) TF (term frequency):

$$tf_{t,d} = \frac{n_{t,d}}{\sum_{k=1}^{T} n_{k,d}} \tag{6}$$

Where the $n_{t,d}$ represents the total numbers of the tth term appears in the dth passage.

The TF approach is to compute the number of times the "term" occurs in the each "passage" (frequency). However, we can not only through the frequency to determine whether this term is vital or not, due to every passage does not have the same lengths. Thus, for the purpose of dealing with this issue, the TF approach do the normalization at each passage (ratio), so that we can through the ratio (frequency) to determine how important the term in the each passage.

(2) IDF (inverse document frequency):

$$idf_t = \log(\frac{D}{d_t}) \tag{7}$$

Where the D is the total numbers of the passages, the d_t is to calculate the sum of the value term t presents in which passages.

On the basis of the IDF approach, we can recognize when the term occurs in more passages, the term IDF value would be more smaller. That is to say, the smaller IDF value can indicate that this term is possible useless in the passages.

The TF-IDF core conception is to multiply the TF and the IDF values and receive the "score", which can be the symbol to determine the relevance with two elements.

The score equation can be described as:

$$score_{t,d} = tf_{t,d} * idf_t \tag{8}$$

We will compute the TF-IDF values at each query iteration through the information stored in the dictionary of the class *Query _ Result* **(Question 2)**, when we execute the Vector Space Model and the BM25 Model.

3.2 Vector Space Model

The main conception in the Vector Space Model is to vectorize the queries and the passages respectively, compute the distance between the two vectors and find the relevance in the end. The correlation will depend on the **Cosine similarity** equation.

The equation can be stated as:

$$Similarity(q,d) = \frac{\sum_{t=1}^{V} q_t * d_t}{\sqrt{\sum_{t=1}^{V} q_t^2} * \sqrt{\sum_{t=1}^{V} d_t^2}} \tag{9}$$

Where the q represents the query vector, the d represents the document vector (passage) and the V represents the vocabulary size.

At the beginning, all the entries in the vector will be assigned the zero value. When the term appears in the document, we will assign the corresponding TF-IDF value from the previous inverted index in both vectors. Nonetheless, at the equation (7), we can notice that when the d_t is 0, which may bring about the mathematical problem.

With the aim of tackling this issue, we adapt the **smooth IDF** equation (10) to replace the equation (7), so that we can avoid having the zeros similarity in the all passages [14].

$$idf_t = 1 + \log(\frac{D}{d_t}) \tag{10}$$

So as to compute the similarity more efficacious and reduce the complexity, at the equation (9), when we calculate the value of the numerator part, especially the vector are pretty sparse, we do not need to focus on the zero elements, in other words, we just need to do the **inner product** to speed up between the non-zero elements.

In the end, we will store the similarity and the pid results in the dictionary *Vector _ Space _ Model*, and write the results in the **VS.txt** file.

3.3 BM25

BM25 is the famous and powerful ranking algorithm based on the binary independence model, regardless of the inter-relationship between the query terms within a document [13]. The BM25 model is also a model that complies with the probabilistic retrieval framework and through the **Score** function to decide the relevance.

As a result, in this section, we will calculate the **Score** between the each query and each passage and determine whether they have relevance with each other.

In line with the course slide **Probabilistic Models**, the Score equation can be depicted as:

$$Score(Q,D) = \sum_{m \in Q} \log(\frac{N - n_m + 0.5}{n_m + 0.5}) * \frac{(k_1+1)f_m}{K + f_m} * \frac{(k_2+1)qf_m}{k_2 + qf_m} \tag{11}$$

The K can be written as:

$$K = k_1((1-b) + b * \frac{dl}{avgdl}) \tag{12}$$

The equation (11) is a simple expression, as a result of we do not have any relevant information in this coursework, so the parameters r and R will be defined as 0. Therefore, we can rewrite the formula as above.

Where the N represents the number of the whole passages, the n_m is the count of the mth term presents in which passages, the f_m represents the frequency of query term m appears in passage D and the qf_m is the count of the term m in query Q. The dl and the $avgdl$ stand for the passage length and the average passage length individually. The k_1, k_2 and b are parameters whose values are set empirically. In this section, we will set the $k_1 = 1.2$, $k_2 = 100$ and $b = 0.75$ advised in the lectures, which is a optimal combination and have the better performance.

During the measure, in the first place, we also need to transform the query and the passage into the vectors, and exploit the information stored in the inverted index. Next, we just need to take these information into the formula to compute the score (similarity). However, we do not store the *avgdl* values in our inverted index, thus, we will record this value during the iteration process. After we finish the iteration (each query) and receive the score of each passage, we will store the scores and the pid results in the dictionary $BM25\ \_\ Model$ and write the results in the $BM25.txt$ file.

3.4 Comparison

Advantage

(1) Vector Space Model:
- Can compute the "scores", thus, the results can be sorted.
- Permit to implement the partical matching.
- Allow to calculate a continuous degree of similarity between queries and documents.
- Simple model based on the linear algebra.

(2) BM25 Model:
- Own the excellent performance and can execute very well in many ad-hoc retrieval tasks.
- On the basis of the equation (11), we can understand that the model can distinguish the short title field and long title field more easily.

Disadvantage

(1) Vector Space Model:
- Unable to resolve the terms which have the similar meaning.
- When the documents are too large, they have poor representation, because of they have the poor similar values.
- Theoretically assumes terms are statistically independent.

(2) BM25 Model:
- The model contains a large number of hacks and heuristics, therefore, the framework of the model is pretty formidable to extend.
- For a given information retrieval measure, it is very challenging to choose the optimal combination of the parameters.

4 RETRIEVAL MODELS, LANGUAGE MODELLING

Language model is a probability distribution over strings of the text which combines the retrieval and indexing into a single model. Besides, the model is not only utilized as an index, but also as an approach of estimating the probability of generating a query. It contains the capacity to extend the framework more readily, and can further to deal with the probabilities of generating the query words themselves from the documents directly [6]. In this section, we will use the query-likelihood model required from the question to finish the following tasks.

The standard query-likelihood model from the lectures can be defined as below:

$$P(Q|M_D) = P(q_1...q_k|M_D) = \prod_{i=1}^{k} P(q_i|M_D) \tag{13}$$

The estimating distribution M_D:

$$M_D = P(word|D) = \frac{f_{word,D}}{|D|} \tag{14}$$

Where the M_D represents that the language model M_D for every document D in the collection. The $q_1 \ldots q_k$ are stand for the each term of the query.

Whereas, the equation (13) contains some obvious drawbacks. First of all, the model cannot offer the notion of the relevance in the model, everything is random sampling. Secondly, if the document do not include all query terms, we will obtain the maximum likelihood estimation equal to zero.

As a result, so as to wrestle with these issues, we will use the smoothing technique for estimating probabilities for missing words and implement the three smoothing methods **(log-probability)** coordinate with the previous inverted structure (information) as follows.

4.1 Laplace smoothing

The core conception of the Laplace smoothing model is that we suppose that we read each term more than once. Accordingly, we will add the one to every count, hence, we can avoid the value equal to the zero. Moreover, we will renormalize the obtain probability. The below equation corresponds to having uniform priors over words.

$$P(word|M_D) = \frac{D(word) + 1}{|D| + |V|} \tag{15}$$

Where $D(word)$ represents the number of $word$ in document D, $|D|$ stands for the count of total terms in document D and $|V|$ is the number of unique words in the entire collection (vocabulary size).

In this section, we will do the same steps as previous questions. We will employ the information stored in the inverted index at first, further to calculate the scores (log value) and record the pid results. After the computation, we will store all the results in the dictionary $Laplace\ \_\ Smoothing\ \_\ Model$, and write the results in the $LM - Laplace.txt$ file in the end.

According to the slide and the previous experiment, we can know that the Laplace smoothing model will provide too much weight to unseen terms and all the unseen terms are smoothed in the same

way. For the purpose of handling the too much weight issue, we can adapt the discounting method "**Lidstone smoothing**" as below.

4.2 Lidstone smoothing

The core conception of the Lidstone smoothing model is to add the small number ϵ to every term count, and further to renormalize the probability. Where the ϵ value is between the 0 and 1. Hence, the model can cope with too much weight to unseen terms issue.

The equation from the lecture can be depicted as:

$$P(word|M_D) = \frac{D(word) + \epsilon}{|D| + \epsilon|V|} \tag{16}$$

In this section, on the basis of the question requirement, we will set the ϵ eqaul to 0.5 at first. Next, we will implement the similar step as the **4.1 sub-question**. Finally, we will store all the results in the dictionary $Lidstone\ \_\ Smoothing\ \_\ Model$, and write the results in the $LM - Lidstone.txt$ file.

Whereas, the Lidstone smoothing model have some demerits. First of all, it treats unseen words equally (add or subtract ϵ). Secondly, it will also encounter the problem that some words are more frequent than others.

4.3 Dirichlet smoothing

Dirichlet smoothing model is a Bayesian smoothing with special prior model and make smoothing depend on sample size. The main conception of the Dirichlet smoothing model is that the term which does not appear in a long document should be assigned low smoothed probability (the longer the document, the lower the probability). Moreover, it possesses the capability to re-estimate the zero probability for the unseen terms by giving them small values derived from the probabilistic values of the seen terms [8].

The formula can be written as:

$$\begin{aligned} P(word|M_D) &= \frac{D(word) + \mu P(word|C)}{|D| + \mu} \\ &= \frac{|D|}{|D| + \mu} P(word|D) + \frac{\mu}{|D| + \mu} P(word|C) \end{aligned} \tag{17}$$

Where C represents the collection and the μ is a constant value.

In this section, in the light of the question requirement, we will set the μ eqaul to 2000 at first. Next, we will implement the similar step as the **4.1 sub-question**. Finally, we will store all the results in the dictionary $Dirichlet\ \_\ Smoothing\ \_\ Model$, and write the results in the $LM - Dirichlet.txt$ file.

According to the slide and the experiment, we can understand that the Dirichlet smoothing model can work wells in the short queries and have better performance. In addition, we also notice that the final sorted results is very similar to the Vector Space Model and the BM25 Model.

REFERENCES

[1] Fondazione Ugo Bordoni. 2018. Information Retrieval Models. https://link.springer.com/referenceworkentry/10.1007%2F978-1-4614-8265-9_916

[2] Latius Hermawan and Maria Bellaniar Ismiati. 2020. Pembelajaran Text Preprocessing berbasis Simulator Untuk Mata Kuliah Information Retrieval. https://journals.usm.ac.id/index.php/transformatika/article/view/1705

[3] Wim Hordijk. 2017. Citizen science: The statistics of language. https://plus.maths.org/content/statistics-language

[4] Chih-Sheng Huang. 2018. Introduction to TF-IDF. https://chih-sheng-huang821.medium.com/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E6%87%89%E7%94%A8-%E5%9E%83%E5%9C%BE%E8%A8%8A%E6%81%AF%E5%81%B5%E6%B8%AC-%E8%88%87-tf-idf%E4%BB%8B%E7%B4%B9-%E5%90%AB%E7%AF%84%E4%BE%8B%E7%A8%8B%E5%BC%8F-2cddc7f7b2c5

[5] Wes McKinney and the Pandas Development Team. 2021. pandas: powerful Python data analysis toolkit Release 1.2.3. https://pandas.pydata.org/pandas-docs/stable/pandas.pdf

[6] Jeremy Pickens. 2000. A Comparison of Language Modeling and Probabilistic Text Information Retrieval Approaches to Monophonic Music Retrieval. http://ismir2000.ismir.net/papers/pickens_paper.pdf

[7] Python Nltk Stemming Tutorial. 2015. https://pythonprogramming.net/stemming-nltk-tutorial/

[8] Abdullah Zawawi Talib Wafa' Za' alAlma' aitah and Mohd Azam Osman. 2019. Structured Dirichlet Smoothing Model for Digital Resource Objects. https://www.researchgate.net/publication/337655207_Structured_Dirichlet_Smoothing_Model_for_Digital_Resource_Objects

[9] Python Nltk Official Website. 2020. https://www.nltk.org/book/ch03.html

[10] Python Nltk Official Website. 2020. https://pythonspot.com/nltk-stop-words/

[11] Python Official Website. 2020. Python Regular Expression. https://www.tutorialspoint.com/python/pdf/python_reg_expressions.pdf

[12] Wikipedia. 2021. Inverted index. https://en.wikipedia.org/wiki/Inverted_index

[13] Wikipedia. 2021. Okapi BM25. https://en.wikipedia.org/wiki/Okapi_BM25

[14] Wikipedia. 2021. tf-idf. https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[15] Łukasz Dębowski. 2000. Zipf's Law: What and Why? https://www.researchgate.net/publication/264873465_Zipf's_Law_What_and_Why