UCL

COMP0084: Information Retrieval and Data Mining - Coursework 1

1

The *Information Retrieval* report explores the field of Information retrieval in language processing. The report reviews various classical and non-classical methods used for retrieving information from textual data in the form of keyword-based search. The report also discusses the preprocessing implemented on the data required for retrieving information efficiently. Furthermore, the report reviews the principle behind different document scoring methods and their impact on document ranking for information retrieval.

1.1 Text Preprocessing

The primary and significant step for efficient information retrieval is text preprocessing, where raw text data is prepared by implementing various cleaning operations. The list of preprocessing steps implemented on the source file for Task 1 is listed below,

- `Lowercase Conversion`
- `Punctuation Removal`
- `Apotrophe Removal`
- `URL Removal`
- `Short-word Removal`
- `Long-word Removal`
- `White-space Removal`
- `Tokenisation`
- `Lemmatisation`
- `Stop-word Removal`

The passages from the source have been cleaned by first converting the entire text to lower case to avoid mismatches due to case sensitivity. All the punctuation marks have been removed along with apostrophes to reduce noise in the data. URLs have been split into seperate words and prefixes like *https* and *www* have been removed to allow hyperlinks to be included in the search results. Words with length shorter than 2 have been removed as these words do not provide any significant information. Words whose length is greater than fifteen have been removed as well under the assumption of not holding any meaning. The data has also been devoid of white-spaces if any.

After performing the first round of preprocessing on raw passages, the passages are then lemmatized. Lemmitization is a process to convert a word to its base form based on the context of the document. The passages are then tokenised, to split them into unigrams that can be accessed individually in the future. The processed unigrams are now collected in the form of a list which is our required Vocabulary for task 1. Although, stop-words will be removed for a test-case in the next part of the task, which is verification of the vocab term frequencies through Zipf's Law.

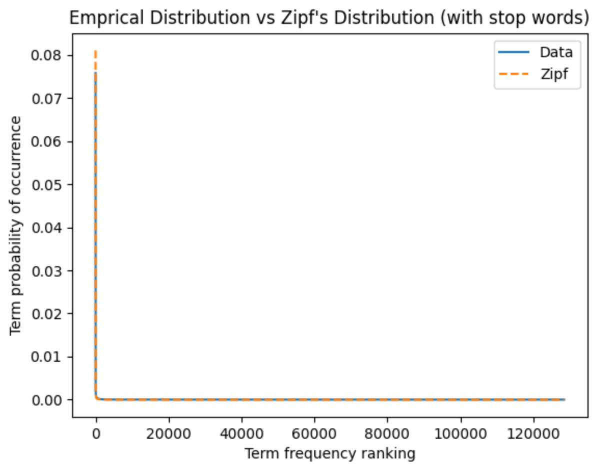

Figure 1 Empirical Distribution vs Zepf's Distribution

1.2 Zipf's Law

Zipf's Law is a statistical tool proposed by American linguist George Kingsley Zipf, that describes the frequencies of observations as inversely proportional to their rank in the frequency distribution, i.e the frequency always follows a rank-frequency distribution. Mathematically,

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

where, k is *Rank of a term*, f is *Normalised Frequency of a term*, N is *No. of terms in vocabulary*, s is *Distribution parameter*. Zipf's distribution for the vocabulary extracted earlier, is derived and plotted against the empirical distribution of the normalised frequencies of terms.

The distribution tends to follow Zipf's distribution decently. However, in the beginning and end of the log-log curve, the empirical distribution deviates from Zipf's distribution. As per Zipf's law (equation 1), $k * f = constant$ *for s=1*, which means when frequency decreases, the rank should increase in a perfectly linear manner. Fundamentally, Zipf's law only takes into account the size of vocabulary *N*, and generates a distribution assuming a corpus of inter-related data and a homogeneous distribution of

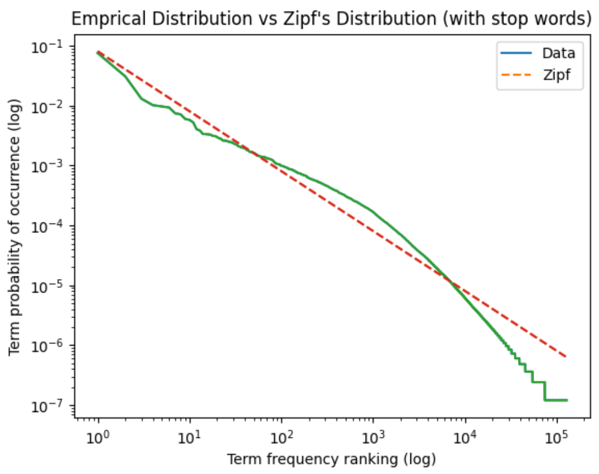

Figure 2 Log-log plot of Empirical Distribution vs Zepf's Distribution

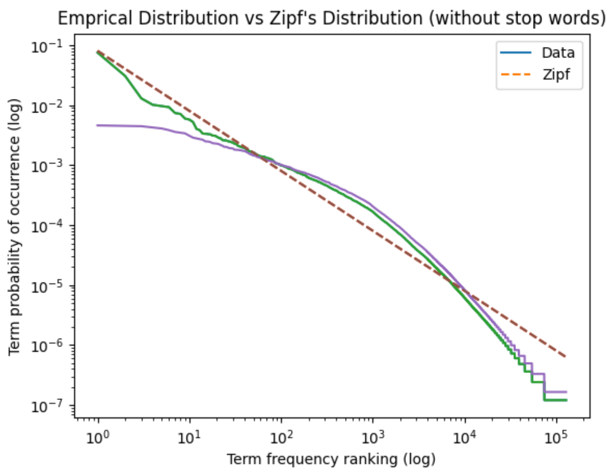

Figure 3 Log-log plot of Empirical Distribution vs Zepf's Distribution - without stop words

more common, less common and rare words. But in reality, probability of occurrence of terms is highly uncertain. The corpus under consideration for the task has a wide variety of unrelated data.It is unlikely that a corpus with non-homogeneous data would follow Zipf's law entirely.

Zipf's Law with stop-words removed When we remove stop-words from the vocab, the empirical distribution deviates more as compared to the case where stop-words were not removed, mainly at the beginning of the plot The reason behind the deviation is that stop words usually are extremely common and hold very high frequencies. When these words are removed from empirical distribution, the distribution of words is not more homogeneous and fails to follow Zipf's law. Zipf's law assumes a homogeneous distribution of common, less common and rare words.

When we statistically compare Zipf's distribution with empirical distribution with and without stop-words, we see that the standard deviation of our empirical distribution is much closer to the standard deviation of Zipf's distribution, as compared to distribution without stop-words. Thus verifies the deviation of the distribution is verified.

$$StdDev(Zipf'sLaw) = 0.0002898 \tag{2}$$

$$StdDev(EmpiricalDistribution) = 0.0002470 \tag{3}$$

$$StdDev((EmpiricalDistribution : no - stopwords) = 6.871 \tag{4}$$

2. Inverted Index

Inverted Index is a data structure used in information retrieval systems that can help optimize the search process for a large collection of text data. In an inverted index, each unique term is indexed with the document IDs it appears in, along with other details like frequency of term in the document, document length etc. that allows efficient searching of terms in the corpus. In this task, the inverted index is built in the form of a dictionary with unique terms of the documents as the dictionary keys. Each of the key(term) contains a value that is a list of tuples. Each tuple contain the pid of a passage the term appears in. It also contains additional information such as the number of occurrences of the term and the length of the passage.

To build the dictionary, the passages are pre-processed and tokenised to get clean keywords and then for each term of the vocabulary, the passages are iterated over row by row using **iterrows()** function, to get a count of the term in each passage. To get the count of the terms in a passage, **nltk.FreqDist()** function is used. All this information gets stored in the dictionary one by one. The process is repeated until all the terms have the list of the passages they appear in. The inverted index dictionary will be useful for future tasks that involve term specific information like the total no. of documents the term appears etc. Stop-words have been removed from the vocabulary because stop-words do not provide any useful information and inturn impact the weightage of terms.

3.1 Vector Space Model

Vector Space Models are methods that are based on representing documents and queries as vectors in a high-dimensional space, where unique terms are the axis and hold a numerical value (weight) corresponding to the query and the documents under consideration. In response to a query, the documents in the collection are ranked using a similarity metric and the highest ranked document is considered as the most relevant one. One of the disadvantage of using vector space models is the lack of semantic consideration and sparsity that it brings along. For a large corpus of data, the method is not very memory efficient.

TF-IDF is one of the classical term weighting methods used in information retrieval. The technique comrises of two components, TF or *Term Frequency* and IDF or *Inverse Document Frequency*. TF-IDF assigns weights to each term in a document based on it's frequency in the document and it's rarity across the collection of documents. The term that occurs frequently in a document but is also rare, i.e. it does not appear in too many documents is likely to be more important than it's counterparts that are more common across the collection.

$$tfidf = tf * idf \tag{5}$$

Cosine Similarity is a measure that calculates the similarity between two vectors in terms of the cosine of angle between them in the vector space. It is calculated as,

$$cosine = \frac{\vec{a}.\vec{b}}{|a||b|} \tag{6}$$

The query passage collection has been preprocessed and ranked on the basis of their cosine similarity score, which has been calculated using vectors of tfidf weights of both query and the document. The documents holding the highest cosine similarity

3.2 BM25

It is a probabilistic model that helps retrieving information by considering multiple factors like relevance, term frequencies and average document lengths etc. Unlike TF-IDF, BM25 also considers the average document length of the collection, so that effect of individual document length is diluted. It has control parameters like k1 and b that control the impact of term frequency and document length on information retrieval. This ensures, no extra weightage goes to documents that have high occurrences of a term or are lengthier than average. The BM25 score between a query and Document is given as,

$$score(q, D) = \sum_{i=1}^{N} IDF(q_i) \frac{f(q_i, D)(k1+1)}{f(q_i, D) + k1(1 - b + b(\frac{|D|}{avgdl})} \tag{7}$$

4.1 Query Likelihood Language Models

Language Models is a probability distribution of words over a document. Query Likelihood Language Model approximates the likelihood of a query q containing the term, given a document D. In order to get wider probability distribution.

Laplace Smoothing

It is an additive smoothing method that takes into consideration the probabilities of unseen words by adding an extra count of 1 every term's probability. But it is done under the assumption that these unseen terms have a uniform prior distribution, which is not a great approximation of unseen data. Hence, it is not suited well to deal with uncertainty.

The Laplace estimates for terms are given by,

$$\frac{(m_1+1)}{|D||V|}, \frac{(m_2+1)}{|D||V|} + \cdots + \frac{(m_V+1)}{|D||V|} \tag{8}$$

Lidstone Smoothing

It is a correction to Laplace Smoothing which involves replacing the addition of fixed count of 1 by a correction parameter epsilon where,$0 < \epsilon <= 1$

Assigning low values to the correction parameter assigns non-zero, but low probabilities to unseen terms which helps avoid overfitting to some extent. The Lidstone Estimates of probability distribution is given by,

$$P(w|D) = \frac{tf_{wD} + \epsilon}{|D| + \epsilon|V|} \tag{9}$$

Dirichlet Smoothing

It approximates a prior distribution of unseen terms through their occurrences in the entire collection of documents which makes the estimation closer to reality. Hence, it acts as an upgrade to the previous smoothing methods that assume a uniform prior distribution of unseen terms.

The Dirichlet distribution is given by,

$$P(w|D) = \frac{tf_{wD} + \mu P(w|C)}{|D| + \mu} \tag{10}$$

4.2 Model Comparison

4.2.1 Which model do you expect to work better?

It is expected that Dirichlet Smoothing will perform better than Laplace or Lidstone Smoothing most of the times because while estimating the probability distribution of unseen terms, Dirichlet Smoothing uses a more concrete way of approximating the probabilities of unseen words, than just assuming the same probability for all unseen terms. This is done by checking how likely the unseen term (unseen in a particular document but available in the larger collection) is on the basis of how many times it has occurred in the entire collection. Dirichlet smoothing allows for the incorporation of prior knowledge about the distribution of the parameters, through the use of the Dirichlet distribution as the prior. This can result in more accurate and meaningful probability estimates, especially when there is some prior knowledge about the distribution of the data. Empirical evidence on few queries:

- For **qid** = 4948358691, we compare two passages, **pid** = 4254836 and **pid** = 6660619.

  query -> `[sensibilities, definition]`,

  pid (4254836) -> `Counter( 'definition': 19, 'russian': 1, 'french': 1, 'japanese': 1, 'vietnamese': 1, 'greek': 1, 'polish': 1, 'turkish': 1, 'portuguese': 1, 'hindi': 1, 'swedish': 1, 'arabic': 1, 'chinese': 1, 'dutch': 1, 'hebrew': 1, 'german': 1, 'korean': 1, 'italian': 1, 'spanish': 1, 'thai': 1)`

  pid (6660619) -> `Counter ( 'estimate': 1, 'papa': 1, 'sensibilities': 3, 'good': 1, 'reason': 1, 'invalidated': 1, 'feelings': 1, 'tender': 1, 'deep': 1, 'ecstatic': 1, 'solemnity': 1, 'swore': 1, 'faith': 1, 'ordinary': 1, 'sex': 1, 'seemed': 1, 'flippancy': 1)`.

  Laplace and Lidstone both score `pid=4254836` higher than `pid=6660619`. This is because both Laplace and Lidstone give more weight to `definition` which occurs 19 times in `pid=4254836`, but Dirichilet model gives more weight to `sensibilities` which occur 3 times in `pid=6660619`. This is better because `len(inverted_index_dict['definition']) == 8691` whereas `len(inverted_index_dict['sensibilities']) == 216` which means that `sensibilities` is a much rarer word in the corpus and should be given more weight during retrieval.

4.2.2 Which models are expected to be more similar?

Laplace and Lidstone techniques are expected to work similar because fundamentally both assume the probabilities of all unseen terms as same. The only difference is that Laplace adds an extra count of 1 to each unseen term, whereas Lidstone does this by adding an additional parameter ϵ, which is a constant whose value can range from 0 to 1. An empiral evidence to support the thesis is given in below tables. The tables reperesent top 5 passages for `qid=719381`. It is noted that Laplace and Lidsto SECURITY WARNING: don't run with debug turned on in production! ne models identify identical passages whereas dirichlet model identifies a different set of passages for the same query.

Laplace Model		
qid	pid	score
719381	1234558	-8.6153
719381	5117472	-9.4360
719381	6262401	-9.4361
719381	7899946	-9.4363
719381	5512425	-9.4364

Table 1 Top 5 passages for $qid = 719381$ with Laplace model

Lidstone Model		
qid	pid	score
719381	1234558	-6.3542
719381	5117472	-7.2214
719381	6262401	-7.2224
719381	7899946	-7.2239
719381	5512425	-7.2246

Table 2 Top 5 passages for $qid = 719381$ with Lidstone model

Dirichlet Model		
qid	pid	score
719381	1234558	-1.6146
719381	4588267	-2.1129
719381	4588270	-2.1129
719381	5117472	-2.1323
719381	7132597	-2.1441

Table 3 Top 5 passages for $qid = 719381$ with Dirichlet model

4.2.3 Choice of ϵ = 0.1 for Lidstone Smoothing

The value of ϵ is a corrective parameter that assigns non-zero probabilities to unseen terms. But since Lidstone is an additive Smoothing, even a small change in the value of ϵ would impact the treatment of unseen terms differently. $\epsilon = 0.1$ is a very small correction value, and indicates that we are restricting the probabilities of unseen terms to small values and not giving high importance to them. This way the probabilities of seen words is preserved and the process of smoothing would not impact the relevance of seen words.

4.2.4 Setting μ = 5000 for Dirichlet Smoothing

The value of μ determines the strength of smoothing. For a large value of μ, the model performs high level of smoothing. But it will be useful when the data is limited or has too many uninformative words that should ideally have low relevance(probability). But for data that is large enough and clean, using a high value of μ would negatively impact the retrieval and would end up assigning lower probabilities to relevant words, thus causing loss of information. Thus, an appropriate value of μ is should be chosen depending on the nature of data we have.

As an example, when we run our model with μ = 5000, the model favours larger document length more than smaller document length while ranking them, even when the smaller document has an extremely high frequency for the term. Thus, higher amount of smoothing hampers the retrieval of documents that are smaller in length even when they are more relevant. We can verify the same through below comparison where high μ value evidently supports passages with high document length. In conclusion, extremely high μ can hinder the more relevant documents to get high rank if they are short in length.

Dirichlet Model $\mu = 5000$			
qid	pid	score	passage length
719381	1234558	-4.020	74
719381	5117472	-4.180	40
719381	6262401	-4.183	54
719381	7899946	-4.187	75
719381	5512425	-4.189	84

Table 4 Top 5 passages for $qid = 719381$ when $\mu = 5000$

Dirichlet Model $\mu = 50$			
qid	pid	score	passage length
719381	1234558	-1.614	74
719381	4588267	-2.112	30
719381	4588270	-2.112	30
719381	5117472	-2.132	40
719381	7132597	-2.144	24

Table 5 Top 5 passages for $qid = 719381$ when $\mu = 50$