# Information Retrieval and Data Mining (2024/25) Coursework 1

**Anonymous ACL submission**

## 1 Task 1

The vocabulary of the passage collection was obtained, consisting of unique terms (uni-grams) extracted after preprocessing. Term frequencies were also calculated to assess whether their distribution follows Zipf's Law. The Zipfian Distribution is given by:

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

Where $k$ is the term's frequency rank, $s$ is the distribution exponent, $N$ is the vocabulary size, and the denominator is the normalisation factor.

Importantly, Zipf's law states that $s$ is equal to 1 in the Zipfian Distribution (Newman, 2005). Before analysing term frequencies to assess whether this corpus of text follows Zipf's Law, preprocessing steps were applied using a custom text processor to extract and format tokens.

1. **Converting all raw text to lowercase:** Converts all text to lowercase for consistency.

2. **Removing punctuation/ all non-alphanumeric characters:** Eliminates characters irrelevant to Zipfian analysis and retrieval models. The removed set includes: `!"#$%&'()*+,-./:;<=>?@[\]-‘{}|~`..

3. **Splitting text with consecutive alphanumerical characters:** Separates mixed alphanumeric sequences (e.g., "al27bn" → ["al", "27", "bn"]).

4. **Stemming:** Reduces words to their root form (e.g., "running" → "run"). In this task, the Porter Stemming algorithm was used to achieve this. The implementation of this was accessed through the Python package gensim. Stemming reduces the number of unique words that need to be processed by the upcoming models in task 3 and 4, improving the computational efficiency of these models.

5. **Tokenise text by whitespace:** Splits text into individual words (1-grams) using whitespace.

6. **Removing stopwords from the tokens list (optional):** Stop-words such as "the" and "is" are removed from the set of tokens. The set of stop-words used are the NTLK set of english stopwords.

Now, the text has been processed and tokenised using the steps above, the size of the index of terms, the vocabulary, can be reported:

| Stopwords Included | Stopwords not included |
|---|---|
| 94365 | 94238 |

Table 1: Index size of the passage collection with stopwords included and stopwords not included.

The normalised frequency of each term is plotted against its rank to assess adherence to Zipf's Law, given by:

$$f(k) = \frac{C}{k} \quad (2)$$

Where $f(k)$ is the normalised frequency at rank $k$, and C is a constant equal to $\left(\sum_{ni=1}^{N} \frac{1}{i}\right)^{-1}$.

To assess adherence to Zipf's Law, a log-log transformation of (2) is applied, and linear regression is used to estimate the gradient and intercept. The constant $C$ is obtained by exponentiating the intercept. These values are compared to theoretical expectations, both with and without stopwords. A derivation follows for further analysis (Fazio and Modica, 2014).

Taking the natural logarithm of (2):

$$\ln f(k) = \ln\left(\frac{C}{k}\right) \tag{3}$$

$$= \ln C - \ln k. \tag{4}$$

This shows that Zipf's Law follows a linear relationship in log-log space, where $\ln f(k)$ (log normalised frequency) decreases linearly with $\ln k$ (log rank), with a gradient of $-1$.

First, the values of $C$ and the gradient are analysed with stopwords included. The empirical distribution and its log-log plot are shown alongside theoretical Zipf's law and the best-fit lines for each.

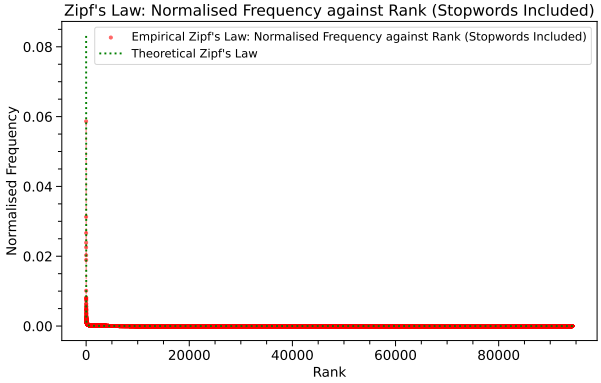


Figure 1: A plot displaying the normalised frequency of each term against the rank of each term. The theoretical distribution of Zipf's law is included.

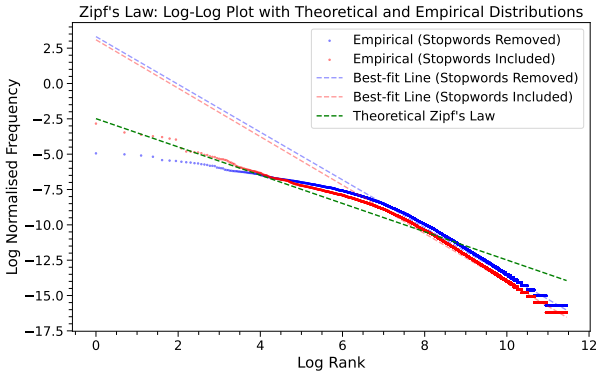


Figure 2: Displays the natural logarithm of normalised term frequency against the natural logarithm of term rank when stop words are included and not included in the analysis. When stop words have been included/ excluded have been shown on the same figure to improve conciseness. A linear regression line of best fit for each is included as well as the theoretical distribution of Zipf's law.

When stop words are included, the calculated $C$ and gradient are 22.1 ± 0.2 and -1.7 ± 0.1, while the theoretical values are 0.08 and -1. This discrepancy suggests the data deviates from Zipf's Law. However, Fig. 2. indicates that linear regression may not be ideal, as high-rank terms skew the fit, leading to an overestimated intercept and underestimated gradient. For log ranks below 8, the distribution qualitatively aligns with Zipf's Law.

Observing the results for when stopwords are excluded below, the calculated values of $C$ and the gradient are 27.6 ± 0.2 and -1.7 ± 0.1 respectively. In addition, observing the distribution in Fig.2., it is clear that the higher rank terms dominate the linear regression fit, resulting in an overestimated intercept and underestimated gradient again. Notably, there is a larger deviation from the theoretical Zipf's distribution from log ranks 6-8.

Linear regression suggests neither distribution strictly follows Zipf's Law, but Fig. 2. shows qualitative adherence in parts. When stop words are included, the empirical distribution aligns with the theoretical one until a hump near log rank 7. This deviation arises from Zipf's Law assuming an infinite vocabulary and no preprocessing (e.g., no stemming). When stop words are included, Zipf's Law is not violated up until the hump; the highest normalised frequency terms are stop words which are taken into account by Zipf's law and these stop words are not stemmed. Therefore, their natural normalised frequency as predicted by Zipf's law is displayed. Moreover, their removal lowers the frequency of the highest-ranked terms as Zipf's law assumes an access to an infinite vocabulary of which stop words are the highest occurring terms.

In both distributions, a hump around log rank 7 is observed. This is suspected to be due to the stemming processing step. There may be a number of words which all have the same root word. This will artificially increase the term count of certain words which will not have the same term count in a natural, un-stemmed corpus of text. The ultimate effect of this is the 'hump' in the mid ranks which represent root words.

Lastly, the abrupt drop in normalised frequency in the tail of both distributions, which skews the linear regression results, can be attributed to the invalid assumption of an infinite vocabulary (Mandelbrot, 1953). Zipf's law assumes an infinitely large corpus, but real-world datasets are finite. In an ideal Zipfian distribution, there should be many instances of extremely rare words. But in this real corpus, there is not enough text and so the vocabulary is limited, which means not enough words are

observed to match the power law desired. This is an effect which is present irrespective of whether stop words are included or not which explains why this effect is observed in both distributions.

To quantify the difference between the distributions, the least squares error including and excluding stopwords is 8.7e-04 and 8.2e-03, respectively. As seen in Fig. 2, the empirical distribution excluding stopwords deviates from Zipf's Law at low and mid ranks. While the stopword-included distribution aligns more closely with Zipf's Law, stopwords will be removed henceforth to improve efficiency and avoid skewing retrieval and query likelihood models.

## 2 Task 2

An inverted index is a data structure used to map words (also defined as terms) to the passages in which they appear. For this task I extracted each unique instance of $PID$ and its corresponding passage from $candidate - passages - top1000.tsv$ and built an inverted index class which extracts a term in the vocabulary, the passages in which they appeared, and the frequency of the term in each passage. The exact dictionary data structure of the inverted index is:

```
inverted_index = {
    "term_1": {pid_1: tf_1, ...,
       pid_n: tf_n},
    "term_2": {pid_3: tf_3, ...,
       pid_m: tf_m},
    ...,
    "term_k": {pid_x: tf_x,...,
       pid_z: tf_z}
}
```

Where each `term_i` represents a unique word in the corpus, `pid_j` is a passage identifier, and `tf_k` is the frequency of the term in that passage.

In addition to this, I have calculated the length of each passage, where the length of each passage is defined as the number of tokens. This is stored in a separate dictionary with the structure:

```
passage_lengths =
    {pid_1: len_1, ..., pid_i:
       len_i}
```

Where `pid_i` is a passage identifier, and `len_i` is the length of each passage.

These quantities have been stored for later tasks. For example, the inverted index is needed for TF-IDF, BM25, and Query Likelihood models because of its fast retrieval of term frequencies. The passage lengths are also needed for the tasks involving BM25 and query likelihood models.

## 3 Task 4

**Which language model do you expect to work better and why? Give a few empirical examples based on your data and experiments.**

Dirichlet smoothing is expected to outperform Laplace and Lidstone smoothing. While all three methods adjust for unseen terms, Dirichlet balances passage-based probabilities with corpus-wide statistics, making it more robust to term rarity. In contrast, Laplace and Lidstone smoothing disproportionately favor high-frequency terms, as they do not consider corpus-wide term distributions. This distinction is clearer when examining their formulas, starting with Laplace smoothing:

$$\log P(Q \mid D) = \sum_{w \in Q} \log \left( \frac{\text{TF}(w, D) + 1}{|D| + V} \right) \quad (5)$$

Where $\text{TF}(w, D)$ is the term frequency of $w$ in document $D$, $|D|$ is the document length, and $V$ is the vocabulary size.

Lidstone smoothing introduces a small additive parameter $\epsilon$, mitigating the over-smoothing of rare words but still favouring frequent terms:

$$\log P(Q \mid D) = \sum_{w \in Q} \log \left( \frac{\text{TF}(w, D) + \epsilon}{|D| + \epsilon V} \right) \quad (6)$$

Where $(0 < \epsilon \leq 1)$.

Dirichlet smoothing, however, dynamically adjusts based on both passage-based and corpus-wide probabilities:

$$\log P(Q \mid D) = \sum_{w \in Q} \log \left( \frac{\text{TF}(w, D) + \mu P(w \mid C)}{|D| + \mu} \right) \quad (7)$$

$$P(w \mid C) = \frac{\sum_D \text{TF}(w, D)}{\sum_D |D|} \quad (8)$$

Where $P(w \mid C)$ is the collection-wide probability of $w$ and $\mu$ controls corpus-wide smoothing.

Observing (5), when a term is missing from a passage ($\text{TF}(w, D) = 0$), Laplace smoothing assigns a uniform probability $P(w \mid D) = \frac{1}{|D|+V}$, artificially inflating the likelihood of all rare words and favoring frequent words in ranking. Lidstone smoothing (6) effect by adjusting with $\epsilon$, but it still

does not account for corpus-wide term distributions.

Dirichlet smoothing (7) solves this issue by down-weighting frequent words and preserving the importance of rare words when they appear in a passage. Unlike Laplace and Lidstone, Dirichlet smoothing ensures rare words do not receive uniform smoothing unless they occur in the passage, making retrieval results more contextually relevant. The smoothing parameter $\mu$ balances the reliance on local document statistics vs. global corpus statistics, leading to fairer passage rankings.

Examining Table 2, Dirichlet smoothing outperforms Laplace and Lidstone smoothing. The top-ranked passages retrieved for the query "function of malt" clearly contain relevant content related to “malt”, ensuring contextual alignment. In contrast, Laplace smoothing completely fails—none of the top passages discuss “malt”; instead, they overemphasize the frequent term “function”. Lidstone smoothing performs slightly better, with the highest-ranked passage capturing the phrase "function of malt", but the 4th and 5th ranked passages lack any relevance to “malt”, demonstrating its limitations.

**Which language models are expected to be more similar and why? Give a few empirical examples based on your data and experiments.**

Laplace and Lidstone smoothing behave similarly, while Dirichlet differs significantly by incorporating corpus-wide term frequencies. Laplace adds 1 to all term counts, and Lidstone adds $\epsilon$, whereas Dirichlet adjusts smoothing based on overall term frequency. As a result, Dirichlet better preserves rare but important terms, while Laplace and Lidstone overemphasize common terms, failing to recognize rare yet meaningful words in a query.

This effect is evident in Table 3. For the query “what type of work is barbering”, “barbering” is the rarest but most important term. Dirichlet correctly prioritizes it, retrieving passages related to barbering, while Laplace and Lidstone focus on common words like “work”, returning general job-related passages instead.

One potential downside of Dirichlet smoothing is its sensitivity to rare terms, which may lead to issues with misspellings. For instance, the query “wat is dopamine” would emphasize “wat” over “dopamine”, whereas Laplace and Lidstone would likely return more relevant results.

**Comment on the value of $\epsilon$ = 0.1 in the Lidstone correction. Is this a good choice? Would there be a better setting (if so, please provide a range of values), and why?**

The choice $\epsilon = 0.1$ is good to the extent that it empirically outperforms Laplace smoothing in identifying relevant passages (Tables 2 and 3). To assess its optimality, I propose a Bayesian estimation method (Zhai and Lafferty, 2004), where the likelihood of observing the term frequency set $\mathbf{f}$ (word counts in D) is:

$$P(\mathbf{f}|\epsilon) = \prod_{w \in V} P(w|D,\epsilon)^{f(w,D)} \tag{9}$$

Where each term probability is computed using Lidstone smoothing. Taking the log-likelihood and defining the prior to be a gamma distribution as $\epsilon >$ 0:

$$P(\epsilon) = \frac{\beta^\alpha}{\Gamma(\alpha)} \epsilon^{\alpha-1} e^{-\beta\epsilon} \tag{10}$$

Where $\alpha$ and $\beta$ are set to 2 and 5 respectively.

Bayes Rule is then applied and the optimal $\epsilon$ is found by maximising the log-posterior. This is done via gradient descent methods.

$$P(\epsilon|\mathbf{f}) \propto P(\mathbf{f}|\epsilon)P(\epsilon) \tag{11}$$

$$\frac{d}{d\epsilon} \log P(\epsilon|\mathbf{f}) = 0 \tag{12}$$

Using this method, it is found that the optimal $\epsilon$ is in fact lower than 0.1, with the optimal value being 0.00077. A common heuristic value for the $\epsilon$ value is the reciprocal of the vocabulary size. This is equal to 0.0000106. Therefore, a suitable, qualitative range for $\epsilon$ is 0.001 - 0.00001.

**If we set μ = 5000 in Dirichlet smoothing, would this be a more appropriate value, and why?**

Broadly speaking, larger values of μ ensures the query-likelihood language model rely heavier on collection-level probabilities, while smaller values of μ ensure the model focuses on document-specific frequencies. In addition, a general heuristic rule of thumb is to set μ equal to the average length of passage lengths, which is 35.9 tokens. Observing Table 2 and 3, where Dirichlet smoothing does a good job at identifying the necessary passages associated with the query, and given this value is close to the heuristic value, it would not be appropriate to set μ = 5000.

Table 2: Top 5 Passages for query: **"function of malt"** (Query ID: 1134787. Tokens: ['function', 'malt'] - stop words have been removed.

| Rank | Laplace Smoothing | Lidstone Smoothing | Dirichlet Smoothing |
|---|---|---|---|
| 1 | Function Handles. Variables that allow you to invoke a function indirectly. A function handle is a data type that stores an association to a function...) | Mucosal-associated Lymphoid Tissue (MALT) refers to the unencapsulated lymphoid tissue distributed in association with mucosal tissues, primarily within their lamina propria. Histology The key feature of MALT... | Mucosal-associated Lymphoid Tissue (MALT) refers to the unencapsulated lymphoid tissue distributed in association with mucosal tissues, primarily within their lamina propria. Histology The key feature of MALT... |
| 2 | A function handle is a MATLAB ® data type that stores an association to a function. Indirectly calling a function enables you to invoke the function regardless... | There is a brisk drying off, followed by a steady kilning that kills the germination process, and curtails any more enzyme activity because enzymes need moisture to function... | There is a brisk drying off, followed by a steady kilning that kills the germination process, and curtails any more enzyme activity because enzymes need moisture to function... |
| 3 | Trigonometric functions are used to calculate the internal angles of triangles. Trigonometric functions are sine function, cosine function, tangent function, cotangent function... | Link to: 23.1 The GI tract 9. This layer of the GI tract is composed of areolar connective tissue that binds the mucosa to the muscularis. A) Submucosa B) Lamina propria C) Epithelium D) Serosa E)... | There is little functional difference between a hoppy amber and a red IPA. I submit, however, that there is a fundamental difference between a session IPA and a hoppy APA, driven primarily by ABV, malt presence, and mouthfeel... |
| 4 | Note: If the original function is a one-to-one function, the inverse will be a function. Definition: The inverse of a function is the set of ordered pairs obtained by interchanging... | Function Handles. Variables that allow you to invoke a function indirectly. A function handle is a data type that stores an association to a function. For example, you can use a function handle... | A major function of the lymphatic system is _____. Aggregates of lymphoid nodules located in the wall of the ileum of the small intestine are _____. Collections of lymphoid tissue (MALT)... |
| 5 | Concave Function. In mathematics, concave function and convex function play an important role. Both the functions are related to each other. A concave function is the negative of a... | A function handle is a MATLAB ® data type that stores an association to a function. Indirectly calling a function enables you to invoke the function regardless of where you call it from... | Link to: 23.1 The GI tract 9. This layer of the GI tract is composed of areolar connective tissue that binds the mucosa to the muscularis. A) Submucosa B) Lamina propria C) Epithelium D) Serosa E)... |

Table 3: Top 5 Passages for query: **"what type of work is barbering"** (Query ID: 1107988. Tokens: ['type','work', 'barber'] - stop words have been removed. Only the stemmed version of 'barbering' is highlighted.

| Rank | Laplace Smoothing | Lidstone Smoothing | Dirichlet Smoothing |
|---|---|---|---|
| 1 | It is often difficult to find work alongside a qualified Clinical Psychologist until you have graduated. You should think about the type of environment that Clinical Psychologists work in, and the type of people they work with... | It is often difficult to find work alongside a qualified Clinical Psychologist until you have graduated. You should think about the type of environment that Clinical Psychologists work in, and the type of people they work with... | Texas offers the following license types under the Barbering program: 1 Class A Barber: 1,500 hours (1,000 high school) 2 Barber Instructor: 3 a course consisting of 750 hours of instruction in barber courses... |
| 2 | How many hours may minors work in a week? A. Sixteen-and 17-year-olds may work up to 44 hours per week. Fourteen-and 15-year-olds may work at most 18 hours per week during the school year and 40 hours per week... | How many hours may minors work in a week? A. Sixteen-and 17-year-olds may work up to 44 hours per week. Fourteen-and 15-year-olds may work at most 18 hours per week during the school year and 40 hours per week... | How to Be a Barber. Three Parts: Evaluating Yourself and the Profession Meeting Education Requirements Working as a Barber Community... |
| 3 | How many hours may minors work in a week? A. Sixteen-and 17-year-olds may work up to 44 hours per week. Fourteen-and 15-year-olds may work at most 18 hours per week during the school year and 40 hours per week... | How many hours may minors work in a week? A. Sixteen-and 17-year-olds may work up to 44 hours per week. Fourteen-and 15-year-olds may work at most 18 hours per week during the school year and 40 hours per week... | As of 2013, three populous states reported more than 1,000 barbering jobs each, according to the statistics bureau. Texas led with 2,140 jobs and average pay of $29,580 per year... |
| 4 | Type I muscle fiber is also known as slow twitch oxidative fibers. Muscle fiber types can be broken down into two main types: slow twitch (Type I) muscle fibers and fast twitch (Type II) muscle fibers.Type I fibers are... | Hello, So I have a DELL desktop computer (Windows XP), and the keyboard won't type anything (obviously, I am using another computer at the moment). I know it still works... | The California Board of Barbering and Cosmetology requires the following school hours, depending on your license type. The Board does not accept Apprentice hours.. Cosmetologist = 1600 hours, Barber = 1500 hours... |
| 5 | Hello, So I have a DELL desktop computer (Windows XP), and the keyboard won't type anything (obviously, I am using another computer at the moment). I know it still works... | People of the same personality type working together in a job create a work environment that fits their type. 2 For example, when Artistic persons are together on a job, they create a work... | The California Board of Barbering and Cosmetology requires the following school hours, depending on your license type. The Board does not accept Apprentice hours.. Cosmetologist = 1600 hours, Barber = 1500 hours... |

## References


Giorgio Fazio and Marco Modica. 2014. Zipf's law for cities: Estimation of regression function parameters. *Bulletin of Geography. Socio-economic Series*, 26:149–159.

Benoit Mandelbrot. 1953. An informational theory of the statistical structure of language. *Communications Theory*, page 486–502.

M. E. J. Newman. 2005. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46(5):323–351.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214.