Information Retrieval and Data Mining (COMP0084)
Coursework 1

Anonymous ACL submission

1 Task 1

The passages in ***passage-collection.txt*** undergo a series of data preprocessing steps, and a dedicated function named **data_pre_processing** is implemented for this purpose to facilitate future use. The data preprocessing procedure for this dataset consists of three main steps:

- **Convert uppercase letters to lowercase:**

  First, the **.lower()** function is used to convert all uppercase words in the passages to lowercase (although in the actual code implementation, the lowercase conversion command was not explicitly included in the function). For example, if the identified term is "The", it would be converted to "the".

- **Tokenization:**

  Furthermore, the **re** library is imported. In this study, to perform tokenization and feature extraction on the textual data, we applied the regular expression `"re.findall(r'\b\w+\b',text)"` to tokenize the text. This regular expression leverages word boundaries `"\b"` and word character `"\w"` matching rules to identify and extract all complete words in the text, including those composed of letters, digits, or underscores, while effectively excluding punctuation marks and whitespace characters. This process provides a standardized vocabulary list consisting of **1-gram**, which serves as the basis for subsequent text analysis and feature construction. For example, given a passage such as "it's apple.", after processing, it would be tokenized into "it", "s", and "apple".

- **Stemming:**

  Finally, the **PorterStemmer** class from Python's **nltk** library is imported, and the Porter Stemmer algorithm is applied to each word to perform stemming, reducing each word to its root form. The process ultimately returns a list of stemmed terms.

In addition, since the subsequent analysis requires examining the impact of stop word removal, this step also filters out stop words from the returned list of word stems. The final size of the identified index of terms after stop word removal is shown in Table 1.

Stop Words	Remove	Not Remove
Size	102083	102210

Table 1: The size of the identified index of terms

Next, the study investigates whether the word frequency distribution in this text dataset follows Zipf's Law. The formula for Zipfian distribution is shown below:

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

In Equation 1, k denotes the term's frequency rank in the corpus, N denotes the total number of terms in the vocabulary, and s is a distribution parameter. When s is set to 1, it corresponds to Zipf's law, which can be expressed by the following formula:

$$f(k; N) = \frac{1}{k \sum_{i=1}^{N} i^{-s}} \tag{2}$$

According to Equation 2, the product of a term's rank and its frequency remains constant($C = \sum_{i=1}^{N} i^{-s}$). Subsequently, the normalised frequency is calculated, and a plot of normalised frequency against frequency ranking is generated, as shown in Figure 1.

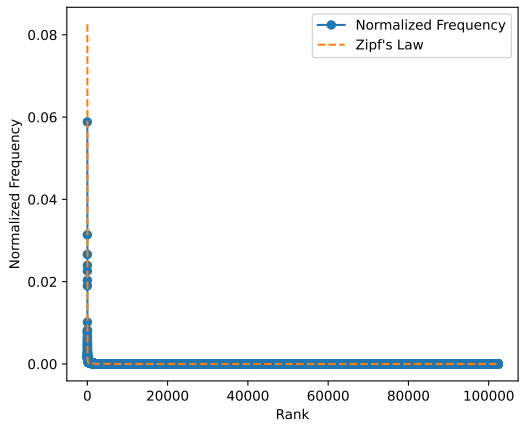

Figure 1: Empirical frequencey against frequency ranking compared with Zipf's law (not remove stop words)

From Figure 1, it can be observed that the curve of the normalised frequency aligns closely with the theoretical line predicted by Zipf's Law, indicating that the terms in this dataset conform to Zipf's Law.

Next, the natural logarithm of both the rank and the normalised frequency is computed, and a log-log plot is generated, as shown in Figure 2.

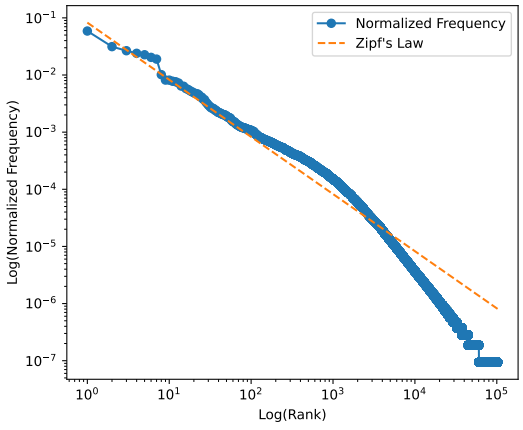

Figure 2: Empirical frequencey against frequency ranking compared with Zipf's law (not remove stop words)-loglog plot

As shown in Figure 2, the empirical word frequency distribution does not perfectly follow the ideal Zipf's law distribution. This deviation is particularly noticeable in the low-frequency (long-tail) region, specifically for terms ranked between $10^{3.5}$ and 10^{5}. In this range, the empirical frequency is significantly lower than the values predicted by Zipf's law.

One possible reason for this discrepancy is the limited corpus size, which may result in many theoretically existing low-frequency terms not being sampled, thus leading to an insufficient number of low-frequency terms.

According to Equation 1, when $s = 1$, the word frequency f follows an inverse relationship with rank k ($f * k = \frac{1}{C}$), consistent with Zipf's law. If the number of low-frequency terms (high-rank terms) increases, their empirical frequencies will collectively rise, causing the tail section of the empirical distribution to shift upward, aligning more closely with the expected Zipfian curve.

Additionally, Figure 2 also reveals a noticeable bulge in the middle-frequency range, specifically for terms ranked between 10^{2} and $10^{3.5}$. Within this range, the empirical frequencies are consistently higher than the predicted values from Zipf's law. This phenomenon could result from several factors:First, the presence of domain-specific terms or frequently occurring phrases in the corpus. Second, the retention of certain functional words or frequently mentioned named entities (e.g., names of people, places, or organizations). Third, an imbalanced corpus composition, where specific topics or domains are overrepresented.

If the coverage of low-frequency terms is improved and the overrepresented frequencies of middle-frequency terms are reduced, the overall word frequency distribution is expected to better conform to the typical power-law shape predicted by Zipf's law.

Furthermore, a log-log plot is generated after removing stop words, as shown in Figure 3.

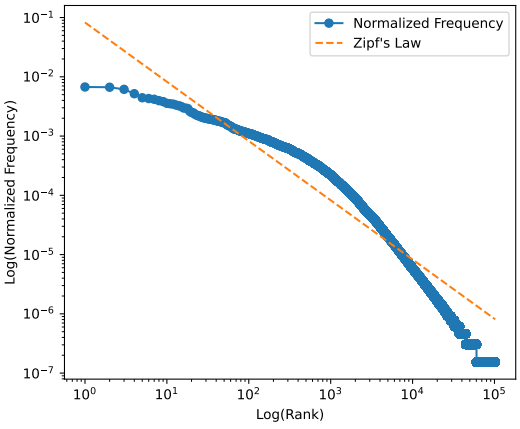

Figure 3: Empirical frequencey against frequency ranking compared with Zipf's law (remove stop words)-loglog plot

Comparing Figure 2 and Figure 3, we observe that after removing stop words, the empirical distribution deviates further from the ideal Zipf's law distribution. This deviation is particularly noticeable in the high-frequency region, where the occurrence probabilities of the top-ranked terms decrease significantly. This phenomenon can be explained as follows.

After removing stop words, terms with lower initial frequencies (higher k move up to occupy the ranks previously held by stop words. As their ranks k decrease, their frequencies f are expected to increase. According to this theoretical shift, removing stop words alone would not directly cause a sharp decline in the occurrence probabilities of high-frequency terms.

However, if the new terms that replace stop words in the top ranks fail to reach the extremely high frequencies of the original stop words, the overall frequency of top-ranked terms will still decrease.

For example, in the original distribution (Figure 2), the top-ranked term "the" had a frequency f of approximately 0.0588. After removing stop words, the new top-ranked term became "1", whose original frequency was only around 0.0041. Although its frequency increased to about 0.0067 after stop word removal, this value is still significantly lower than the original frequency of "the".

This pattern can be generalized to explain the overall downward shift in the frequencies of high-ranked terms observed in Figure 3.

2 Task 2

2.1 Whether to remove stop words

According to the results obtained in Task 1, it can be concluded that removing stop words causes the data to no longer follow Zipf's law, which may disrupt the completeness of the Zipfian distribution. In addition, in Task 3, the IDF calculation penalizes high-frequency terms, serving a similar function to stop word removal. Furthermore, there are certain scenarios where stop words can actually help distinguish passages that contain them. For example, if the query is "the apple", and the two passages to be retrieved are "the apple juice" and "apple", removing the stopword "the" would make it impossible for the query to differentiate between these two passages.

Therefore, in this section, stop words will be retained when constructing the inverted index.

2.2 How to build inverted index

The following part will describe the method used to build the inverted index. First, the vocabulary table obtained from Task 1 is loaded, and the number of passages in the file ***candidate-passages-top1000.tsv*** is counted and denoted as N.

Next, an empty dictionary is created using the command:

inverted_index = collections.defaultdict(dict)

This dictionary will be used to store the inverted index information. The file ***candidate-passages-top1000.tsv*** is then read line by line to extract the pid and passsage information. Each passage is converted to lowercase using the **.lower()** method, followed by calling the **data_pre_processing** function to preprocess the passage. During preprocessing, only terms that exist in the vocabulary are retained, ensuring that the constructed inverted index is vocabulary-based.

Furthermore, the **collections.Counter** function is used to calculate the term frequency (tf) for each word within the passage. Finally, the term frequency, along with the corresponding pid, is stored into the inverted index dictionary.

The final storage format of the inverted index is:

{"word" : {"pid" : tf}}

2.3 Calculate inverse document frequency

In addition, the inverse document frequency (IDF) is also calculated in this task, which will be used in Task 3 for computing cosine similarity. The formula for calculating IDF is as follows:

$$\text{IDF}_t = \log_{10}\left(\frac{N}{n_t}\right) \tag{3}$$

In Equation 3, N represents the number of documents in the collection, which has already been calculated in Section 2.2. n_t denotes the number of documents in which term t appears. The computed IDF values, along with the inverted index, are saved into a JSON file for future use.

3 Task 3

In this task, two files and the inverted index obtained in Task 2 are loaded. For each query provided in ***test-queries.tsv***, the relevance scores between the query and candidate passages are computed using cosine similarity based on TF-IDF vectors and the BM25 method, respectively. Based on these computed scores, the top 100 candidate passages for each query are selected.

4 Task 4

In this task, three types of query likelihood language models with smoothing were implemented using ***test-queries.tsv*** and ***candidate-passages-top1000.tsv***. The log-probability of each passage is then computed to rank and retrieve relevant passages for each query.

The first model applies **Laplace smoothing**, and its formula is as follows:

$$P(w \mid D) = \frac{\text{TF}(w, D) + 1}{|D| + |V|} \tag{4}$$

where $\text{TF}(w, D)$ denotes the term frequency of word w in passage D, $|D|$ denotes the total number of words in passage D, and $|V|$ denotes the size of the vocabulary.

The second model applies **Lidstone correction**, and its formula is given by:

$$P(w \mid D) = \frac{\text{TF}(w, D) + \epsilon}{|D| + \epsilon \cdot |V|} \tag{5}$$

where ϵ denotes a small positive smoothing parameter, controlling the amount of probability mass assigned to unseen terms.

The third model applies **Dirichlet smoothing** in the query likelihood language model. The formula used is as follows:

$$P(w \mid D) = \frac{|\text{D}|}{|\text{D}| + \mu} \cdot \frac{\text{TF}(w, D)}{|\text{D}|} + \frac{\mu}{|\text{D}| + \mu} \cdot \frac{\text{TF}(w, C)}{|C|} \tag{6}$$

Where:

- μ denotes the Dirichlet smoothing parameter.
- $\text{TF}(w, C)$ denotes the total occurrence of term w in the whole collection.
- $|C|$ denotes the total number of terms in the whole collection.

4.1 D11 discussion

1. **Which language model do you expect to work better and why?**

   All three smoothing models aim to address the same fundamental issue: how to handle terms that exist in the vocabulary but do not appear in the current document — assigning them a non-zero probability rather than a zero probability.

   However, both Laplace smoothing and Lidstone correction assign the same fixed probability to all unseen terms, regardless of their importance or frequency in the overall collection. This one-size-fits-all approach overlooks the fact that different terms have different inherent importance. As a result, an extremely rare term and a very common term (both unseen in the current document) would receive the same probability, which is inconsistent with linguistic intuition.

   Dirichlet smoothing, on the other hand, addresses this issue by incorporating information from the overall term distribution in the entire corpus. Specifically, when assigning probabilities to unseen terms, Dirichlet smoothing does not treat all terms equally; instead, it takes into account how important each term is in the global corpus. Therefore, even if a common term does not appear in the current document, it can still receive a relatively higher smoothed probability than a rare term.

   As shown in Table 2, the results obtained by the Laplace smoothing and Lidstone correction are exactly the same. However, compared to the Dirichlet model, both Laplace smoothing and Lidstone correction fail to account for the global importance of the word "mean". In both models, the passages ranked in the last two positions do not contain the word "mean", despite the fact that this word is much rarer than "of" and "heat" in the entire collection. This example clearly highlights the major limitation of the Laplace and Lidstone models: they do not effectively incorporate word frequency information from the entire collection.

2. **Which language models are expected to be more similar and why?**

   The Laplace model and the Lidstone model exhibit strong similarities, as demonstrated in Table 2. According to Table 2, the top 5 passages retrieved by both the Laplace smoothing model and the Lidstone correction model are exactly the same, indicating that the two methods produce highly consistent ranking results. From a mathematical perspective, as shown in Equation 5, Since both methods smooth the frequencies in a similar manner, they can produce consistent ranking results even if the

smoothed frequency values differ. The value range of ϵ is 0 to 1, and its purpose to reduce the frequency assigned to words that do not appear in the passage. The Lidstone model simplifies to the Laplace model when $\epsilon = 1$. Here, ϵ is set to 0.1, resulting in a weaker smoothing effect compared to the Laplace model.

3. **Comment on the value of ϵ = 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?**

   Setting $\epsilon = 0.1$ is not necessarily an optimal choice. The following section will derive how to determine a more appropriate range for ϵ.

   First, the smoothed probability for words that appear in the document is Equation 5. The smoothed probability of a word not appearing in the document is:

$$P(w \mid D) = \frac{\epsilon}{|D| + \epsilon \cdot |V|} \tag{7}$$

   Subsequently, the sum of the smoothed probabilities of the words appearing in the document is calculated as shown in Equation 8.

$$\frac{N_p - N_{up}\epsilon}{N_p + \epsilon N_v} \tag{8}$$

   The synthesis of the smoothed probabilities of words not appearing in the document, as shown in Equation 9.

$$\frac{(N_v - N_{up})\epsilon}{N_p + \epsilon N_v} \tag{9}$$

   In these two Equations, N_p represent of the number of words in vacabulary, N_{up} represent of the number of unique words in passage, N_v represent of the number of words in passage.

   In practice, in order to maintain the amount of information in the document, we would like the total probability of an actual occurrence of a word to be much greater than the total probability of a non-occurrence of a word, as shown in Equation 10.

$$\frac{(N_v - N_{up})\epsilon}{N_p + \epsilon N_v} << \frac{N_p - N_{up}\epsilon}{N_p + \epsilon N_v} \tag{10}$$

   Simplifying Equation 10 yields the range of values of eplison as shown in Equation 11.

$$\epsilon << \frac{N_p}{N_v - 2N_{up}} \tag{11}$$

   Estimating N_p and N_{up} using the mean values yields a mean of 58.23 for N_p, 36.9 for N_{up}, and N_v is equal to 102210 in my case, which ultimately yields that ϵ should take a value much less than 0.00057

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

   In terms of parameter selection, setting $\mu = 5000$ is not appropriate. The parameter μ can be interpreted as a form of sampling additional information from the overall collection, meaning that its magnitude should be similar with the average passage length within the corpus. According to the constructed index, the average passage length in the corpus is approximately 58.23. Therefore, the initial choice of $\mu = 50$ better reflects the actual passage length and offers a more reasonable balance between local and global information. Of course, in practical applications, the optimal value of μ should still be determined using a validation set to ensure the best model performance.

Query Text: ['mean', 'of', 'heat', 'capac']		
Rank	**Laplace smoothing and Lidstone smoothing when = 0.1**	**Dirichlet smoothing when = 50**
1	['r8', '3', 'heat', 'capac', 'a', 'constant', 'or', 'mean', 'heat', 'capac', 'heat', 'capac', 'are', 'typic', 'express', 'as', 'a', 'function', 'of', 'temperatur', 'by', 'the', 'quadrat', 'r8', '3', '1', 'we', 'will', 'now', 'consid', 'both', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'and', 'variabl', 'heat', 'capac', 'for', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'r8', '3', '2', 'the', 'circumflex', 'denot', 'that', 'the', 'heat', 'capac', 'are', 'evalu', 'at', 'some', 'mean', 'temperatur', 'valu', 'between', 't', 'r', 'and', 't', 'r8', '3', '3', 'in', 'a', 'similar', 'fashion', 'we', 'can', 'write', 'the', 'integr', 'involv', 'i', 'and', 'in', 'equa', 'tion', 'r8', '3', '1', 'as', 'is', 'the', 'mean', 'heat', 'capac', 'of', 'speci', 'i', 'between', 't', 'i', '0', 'and', 't', 'r8', '3', '4', 'substitut', 'the', 'mean', 'heat', 'capac', 'into', 'equat', 'r8', '3', '1', 'the', 'steadi', 'state']	['r8', '3', 'heat', 'capac', 'a', 'constant', 'or', 'mean', 'heat', 'capac', 'heat', 'capac', 'are', 'typic', 'express', 'as', 'a', 'function', 'of', 'temperatur', 'by', 'the', 'quadrat', 'r8', '3', '1', 'we', 'will', 'now', 'consid', 'both', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'and', 'variabl', 'heat', 'capac', 'for', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'r8', '3', '2', 'the', 'circumflex', 'denot', 'that', 'the', 'heat', 'capac', 'are', 'evalu', 'at', 'some', 'mean', 'temperatur', 'valu', 'between', 't', 'r', 'and', 't', 'r8', '3', '3', 'in', 'a', 'similar', 'fashion', 'we', 'can', 'write', 'the', 'integr', 'involv', 'i', 'and', 'in', 'equa', 'tion', 'r8', '3', '1', 'as', 'is', 'the', 'mean', 'heat', 'capac', 'of', 'speci', 'i', 'between', 't', 'i', '0', 'and', 't']
2	['r8', '3', 'heat', 'capac', 'a', 'constant', 'or', 'mean', 'heat', 'capac', 'heat', 'capac', 'are', 'typic', 'express', 'as', 'a', 'function', 'of', 'temperatur', 'by', 'the', 'quadrat', 'r8', '3', '1', 'we', 'will', 'now', 'consid', 'both', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'and', 'variabl', 'heat', 'capac', 'for', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'r8', '3', '2', 'the', 'circumflex', 'denot', 'that', 'the', 'heat', 'capac', 'are', 'evalu', 'at', 'some', 'mean', 'temperatur', 'valu', 'between', 't', 'r', 'and', 't', 'r8', '3', '3', 'in', 'a', 'similar', 'fashion', 'we', 'can', 'write', 'the', 'integr', 'involv', 'i', 'and', 'in', 'equa', 'tion', 'r8', '3', '1', 'as', 'is', 'the', 'mean', 'heat', 'capac', 'of', 'speci', 'i', 'between', 't', 'i', '0', 'and', 't']	['r8', '3', 'heat', 'capac', 'a', 'constant', 'or', 'mean', 'heat', 'capac', 'heat', 'capac', 'are', 'typic', 'express', 'as', 'a', 'function', 'of', 'temperatur', 'by', 'the', 'quadrat', 'r8', '3', '1', 'we', 'will', 'now', 'consid', 'both', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'and', 'variabl', 'heat', 'capac', 'for', 'the', 'case', 'of', 'constant', 'or', 'mean', 'heat', 'capac', 'r8', '3', '2', 'the', 'circumflex', 'denot', 'that', 'the', 'heat', 'capac', 'are', 'evalu', 'at', 'some', 'mean', 'temperatur', 'valu', 'between', 't', 'r', 'and', 't', 'r8', '3', '3', 'in', 'a', 'similar', 'fashion', 'we', 'can', 'write', 'the', 'integr', 'involv', 'i', 'and', 'in', 'equa', 'tion', 'r8', '3', '1', 'as', 'is', 'the', 'mean', 'heat', 'capac', 'of', 'speci', 'i', 'between', 't', 'i', '0', 'and', 't']
3	['it', 'is', 'the', 'amount', 'of', 'energi', 'requir', 'to', 'rais', 'the', 'mass', 'of', '1', 'gram', 'of', 'a', 'substanc', 'by', '1', 'c', 'q', 'specif', 'heat', 'capac', 'x', 'mass', 'x', 'd', 't', 'the', 'specif', 'heat', 'capac', 'of', 'water', 'is', '4', '184', 'j', 'g', '1', 'k', '1', 'thi', 'mean', 'that', '4', '184', 'j', 'of', 'energi', 'are', 'requir', 'to', 'rais', 'each', 'gram', 'of', 'water', 'by', 'each', 'c', 'heat', 'capac', 'can', 'also', 'be', 'use', 'in', 'calcul', 't', 'is', 'the', 'amount', 'of', 'energi', 'requir', 'to', 'rais', 'the', 'mass', 'of', '1', 'gram', 'of', 'a', 'substanc', 'by', '1', 'c', 'q', 'specif', 'heat', 'capac', 'x', 'mass', 'x', 'd', 't', 'the', 'specif', 'heat', 'capac', 'of', 'water', 'is', '4', '184', 'j', 'g', '1', 'k', '1', 'thi', 'mean', 'that', '4', '184', 'j', 'of', 'energi', 'are', 'requir', 'to', 'rais', 'each', 'gram', 'of', 'water', 'by', 'each', 'c', 'heat', 'capac', 'can', 'also', 'be', 'use', 'in', 'calcul']	['it', 'is', 'the', 'amount', 'of', 'energi', 'requir', 'to', 'rais', 'the', 'mass', 'of', '1', 'gram', 'of', 'a', 'substanc', 'by', '1', 'c', 'q', 'specif', 'heat', 'capac', 'x', 'mass', 'x', 'd', 't', 'the', 'specif', 'heat', 'capac', 'of', 'water', 'is', '4', '184', 'j', 'g', '1', 'k', '1', 'thi', 'mean', 'that', '4', '184', 'j', 'of', 'energi', 'are', 'requir', 'to', 'rais', 'each', 'gram', 'of', 'water', 'by', 'each', 'c', 'heat', 'capac', 'can', 'also', 'be', 'use', 'in', 'calcul', 't', 'is', 'the', 'amount', 'of', 'energi', 'requir', 'to', 'rais', 'the', 'mass', 'of', '1', 'gram', 'of', 'a', 'substanc', 'by', '1', 'c', 'q', 'specif', 'heat', 'capac', 'x', 'mass', 'x', 'd', 't', 'the', 'specif', 'heat', 'capac', 'of', 'water', 'is', '4', '184', 'j', 'g', '1', 'k', '1', 'thi', 'mean', 'that', '4', '184', 'j', 'of', 'energi', 'are', 'requir', 'to', 'rais', 'each', 'gram', 'of', 'water', 'by', 'each', 'c', 'heat', 'capac', 'can', 'also', 'be', 'use', 'in', 'calcul']
4	['the', 'specif', 'heat', 'capac', 'of', 'sea', 'water', 'would', 'be', 'rel', 'high', 'water', 'ha', 'a', 'specif', 'heat', 'of', '3', '9', 'j', 'k', 'g', 'that', 'it', 'take', '3', '9', 'joul', 'of', 'energi', 'to', 'heat', '1', 'gram', 'of', 'water', '1', 'degre', 'celsiu', 'or', 'kelvin', 'metal', 'have', 'a', 'low', 'heat', 'capac', 'therefor', 'it', 'take', 'littl', 'energi', 'to', 'heat', 'up', '1', 'gram', 'of', 'copper', 'for', 'exampl', '0', '39', 'joul', 'to', 'heat', '1', 'g', 'of', 'cu', '1', 'degre', 'the', 'heat', 'capac', 'of', 'land', 'is', 'lower', 'than', 'that', 'of', 'water']	['no', 'the', 'amount', 'of', 'heat', 'a', 'substanc', 'absorb', 'is', 'determin', 'by', 'it', 'heat', 'capac', 'the', 'specif', 'heat', 'capac', 'of', 'sea', 'water', 'would', 'be', 'rel', 'high', 'water', 'ha', 'a', 'specif', 'heat', 'of', '4', '18', 'j', 'k', 'g', 'that', 'mean', 'it', 'take', '4', '18', 'joul', 'of', 'energi', 'to', 'heat', '1', 'gram', 'of', 'water', '1', 'degre', 'celsiu', 'or', 'kelvin', 'metal', 'have', 'a', 'low', 'heat', 'capac', 'therefor', 'it', 'take', 'littl', 'energi', 'to', 'heat', 'up', '1', 'gram', 'of', 'copper', 'for', 'exampl', '0', '39', 'joul', 'to', 'heat', '1', 'g', 'of', 'cu', '1', 'degre', 'the', 'heat', 'capac', 'of', 'land', 'is', 'lower', 'than', 'that', 'of', 'water']
5	['note', 'that', 'the', 'specif', 'heat', 'is', 'per', 'unit', 'mass', 'thu', 'the', 'specif', 'heat', 'of', 'a', 'gallon', 'of', 'milk', 'is', 'equal', 'to', 'the', 'specif', 'heat', 'of', 'a', 'quart', 'of', 'milk', 'a', 'relat', 'quantiti', 'is', 'call', 'the', 'heat', 'capac', 'c', 'of', 'an', 'object', 'the', 'relat', 'between', 's', 'and', 'c', 'is', 'c', 'mass', 'of', 'obect', 'x', 'specif', 'heat', 'of', 'object', 'a', 'tabl', 'of', 'some', 'common', 'specif', 'heat', 'and', 'heat', 'capac', 'is', 'given', 'below', 'consid', 'the', 'specif', 'heat', 'of', 'copper', '0', '385', 'j', 'g', '0c', 'what', 'thi', 'is', 'that', 'it', 'take', '0', '385', 'joul', 'of', 'heat', 'to', 'rais', '1', 'gram', 'of', 'copper', '1', 'degre', 'celciu', 'thu', 'if', 'we', 'take', '1', 'gram', 'of', 'copper', 'at', '25', '0c', 'and', 'add', '1', 'joul', 'of', 'heat', 'to', 'it', 'we', 'will', 'find', 'that', 'the', 'temperatur', 'of', 'the', 'copper', 'will', 'have', 'risen', 'to', '26', '0c']	['specif', 'heat', 'capac', 'formula', 'the', 'specif', 'heat', 'capac', 'of', 'a', 'substanc', 'is', 'the', 'amount', 'of', 'heat', 'requir', 'to', 'rais', 'one', 'gram', 'of', 'the', 'substanc', 'by', 'one', 'degre', 'celsiu', 'water', 'for', 'exampl', 'ha', 'a', 'specif', 'heat', 'capac', 'of', '4', '18', 'thi', 'mean', 'to', 'heat', 'one', 'gram', 'of', 'water', 'by', 'one', 'degre', 'celsiu', 'it', 'would', 'requir', '4', '18', 'joul', 'of', 'energi']

Table 2: The top five query results obtained by the three methods for the query tokens: "mean", "of", "heat", "capac"