
CORPUS Q&A TOOL

COL106 Assignment 7 Report

Authors

Rakshit Sohlot, Arvind Kumar Sharma, Naman Goel, Krishna Sharma
Indian Institute of Technology, Delhi
November, 2023

Contents

1	Ranking Algorithm	3
1.1	Okapi BM25+	3
1.1.1	Laplace Smoothing	3
1.1.2	Dynamics of the score function and reason for not using TF-IDF . . .	4
1.2	Changes made to the ranking function	5
1.3	Change in search results obtained	5
1.3.1	Search results obtained in Part-I	5
1.3.2	Search results obtained in Part-II	6
2	Additional header files	7
3	Use of LLMs other than ChatGPT	7
4	Prompt Engineering	7
4.1	Prompt used for ChatGPT and PaLM	8
4.2	Prompt used for HugChat	8
4.3	Prompt Engineering Paramters	8
5	Optimizations	9
5.1	Finding Top-k Paragraphs	9
5.2	Optimal Value of k in Top-k	9
5.3	Refinement of query	9

1 Ranking Algorithm

1.1 Okapi BM25+

Okapi BM25 (BM is an abbreviation of *best matching*) is a ranking function used by search engines to estimate the relevance of documents to a given search query. It is based on the probabilistic retrieval framework developed in the 1970s and 1980s by Stephen E. Robertson, Karen Spärck Jones, and others. BM25+ is an extension of BM25.[1]

The scoring function is given by:

$$\text{score}(P, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \left[\frac{f(q_i, P) \cdot (k_1 + 1)}{f(q_i, P) + k_1 \cdot \left(1 - b + b \cdot \frac{|P|}{\text{avgdl}}\right)} + \delta \right]$$

where,

- Q is a query which contains the keywords q_1, q_2, \dots, q_n
- P is the paragraph
- $f(q_1, P)$ is the number of times q_1 occurs in the paragraph P
- $|P|$ is the length of the paragraph in words
- avgdl is the average paragraph length in the corpus
- k_1 and b are free parameters. Another free parameter, δ is a tuning parameter introduced to address term frequency saturation. Their values are : $k_1 \in [1.2, 2.0]$, $b = 0.75$ and the default value of δ is 1.0 in the absence of any training data
- $\text{IDF}(q_i)$ is the inverse document frequency weight of the word q_i , usually computed as :

$$\text{IDF}(q_i) = \ln \left(\frac{n - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right)$$

where n is the total number of paragraphs in the corpus and $n(q_i)$ is the total number of paragraphs containing the word q_i

1.1.1 Laplace Smoothing

Laplace smoothing or additive smoothing is a statistical technique to increase the probability of less likely elements by increasing the counts of all the elements in the dataset by a constant.[2]

Laplace smoothing is done in the IDF formula by adding 0.5 in the numerator as well as the denominator of the fraction contained in the term inside \ln . Its purpose is to prevent division by zero while handling zero frequency terms (i.e., terms that do not appear in any document in the collection). Without smoothing, if a term q_i has zero document frequency ($n(q_i) = 0$), the denominator becomes zero, resulting in undefined or infinite values.

1.1.2 Dynamics of the score function and reason for not using TF-IDF

The score function has the same structure as $TF * IDF$, but with refined values of TF and IDF . Some of the refinements done are as follows [3]:

1. Term Saturation

If a paragraph contains 200 occurrences of “elephant”, is it really twice as relevant as a paragraph that contains 100 occurrences? We could argue that if “elephant” occurs a large enough number of times, say 100, the paragraph is almost certainly relevant, and any further mentions don’t really increase the likelihood of relevance. To put it a different way, once a paragraph is saturated with occurrences of a term, more occurrences shouldn’t have a significant impact on the score. So we’d like a way to control the contribution of TF to our score. We’d like this contribution to increase fast when TF is small and then increase more slowly, approaching a limit, as TF gets very big.

This is done by replacing TF with $\frac{TF}{TF+k_1}$, where : TF (or Term Frequency) = $f(q_1, P)$ and k_1 is a free parameter (fixed constant). For a fixed value of k_1 , the value of $\frac{TF}{TF+k_1}$ saturates for large values of TF . It can be seen in the following graph as well for various values of k_1 .

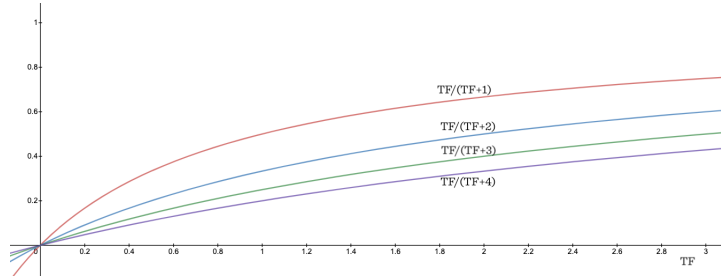


Figure 1: Variation of $\frac{TF}{TF+k_1}$ with TF for various values of k_1

A fortunate side-effect of using $\frac{TF}{TF+k_1}$ to account for term saturation is that we end up rewarding complete matches over partial ones. That’s to say, we reward paragraphs that match more of the terms in a multi-term query over paragraphs that have lots of matches for just one of the terms.

Let’s say that “cat” and “dog” have the same IDF values. If we search for “cat dog”, we’d like a paragraph that contains one instance of each term to do better than a paragraph that has two instances of “cat” and none of “dog.” If we were using raw TF , they’d both get the same score. But let’s do our improved calculation assuming $k = 1$. In our “cat dog” paragraph, “cat” and “dog” each have $TF = 1$, so each is going to contribute, $\frac{TF}{TF+1} = \frac{1}{2}$ to the score, for a total of 1. In our “cat cat” paragraph, “cat” has a TF of 2, so it’s going to contribute, $\frac{TF}{TF+1} = \frac{2}{3}$ to the score. The “cat dog” paragraph wins, because “cat” and “dog” contribute more when each occurs once than “cat” contributes when it occurs twice.

2. Paragraph Length

If a paragraph happens to be really short and it contains “elephant” once, that’s a

good indicator that “elephant” is important to the content. But if the paragraph is really, really long and it mentions “elephant” only once, the paragraph is probably not about elephants. So we’d like to reward matches in short paragraphs while penalizing matches in long paragraphs.

We’ll achieve this by introducing the ratio $\frac{|P|}{avgdl}$ in the denominator. Here, $|P|$ is the length of the paragraph in words, and $avgdl$ is the average paragraph length in the corpus.

When a paragraph is of average length, $\frac{|P|}{avgdl} = 1$, and our multiplier doesn’t affect k_1 at all. For a paragraph that’s shorter than average, we’ll be multiplying k_1 by a value between 0 and 1, thereby reducing it, and increasing $\frac{TF}{TF+k_1}$. For a paragraph that’s longer than average, we’ll be multiplying k_1 by a value greater than 1, thereby increasing it, and reducing $\frac{TF}{TF+k_1}$. The multiplier also puts us on a different TF saturation curve. Shorter paragraphs will approach a TF saturation point more quickly while longer paragraphs will approach it more gradually.

1.2 Changes made to the ranking function

- We have fixed the value of k_1 at 1.5 which lies close to the average of possible values it can take.
- The value of the tuning parameter δ is same as its default value, 1.0.
- Since the value of k_1 has been fixed, we have removed the $k_1 + 1$ term from the numerator since it has equal values for all paragraphs. This is done to make computation faster.
- As per our observation the standard deviation of the length of paragraphs is quite low, so the term $\frac{|P|}{avgdl}$ is very close to 1 for most paragraphs. Hence, we have approximated $(1 - b + b \cdot \frac{|P|}{avgdl}) \approx 1$ to make our code computationally more efficient.

The final formula used in our code is:

$$\text{score}(P, Q) = \sum_{i=1}^n \ln \left(\frac{n - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right) \cdot \left[\frac{f(q_i, P)}{f(q_i, P) + 1.5} + 1.0 \right]$$

1.3 Change in search results obtained

A study of search results obtained for the query “What were the views of Mahatma Gandhi on the Partition of India?”

1.3.1 Search results obtained in Part-I

The top-k paragraphs obtained using the formula provided in Part-I of the assignment,

$\text{score}(p) = \sum_{i=1}^k f_p(w_i) \cdot s(w_i)$, where $s(w) = \frac{\text{frequency}(w) \text{ in specific corpus} + 1}{\text{frequency}(w) \text{ in general corpus} + 1}$, were:

1. Book code: 9 Page: 484 Paragraph: 4
2. Book code: 44 Page: 92 Paragraph: 1
3. Book code: 15 Page: 501 Paragraph: 3

4. Book code: 26 Page: 428 Paragraph: 1

5. Book code: 14 Page: 246 Paragraph: 2

When we fed these paragraphs to ChatGPT, it gave the following response:

“Based on the provided excerpts, there is no direct information about Gandhi’s view on partition. The excerpts cover various topics such as Gandhi’s imprisonment, his family background, negotiations with the government, and his proposal for a reduction in rents. The given excerpts do not discuss Gandhi’s specific views on partition.”

As it is clear from the given response, the scoring algorithm failed to rank those paragraphs higher which actually contained Gandhi’s views on partition.

1.3.2 Search results obtained in Part-II

The top-k paragraphs obtained by using the modified Okapi BM25+ algorithm were:

1. Book code: 96 Page: 67 Paragraph: 3

2. Book code: 31 Page: 285 Paragraph: 1

3. Book code: 10 Page: 250 Paragraph: 4

4. Book code: 95 Page: 55 Paragraph: 6

5. Book code: 15 Page: 513 Paragraph: 1

These paragraphs were highly relevant, since the above-mentioned paragraphs in vol. 95 and 96 had Gandhi’s views on the Partition of India, whereas other paragraphs mentioned had Gandhi’s views on the Partition of India.

When we fed these paragraphs to ChatGPT, it gave the following response:

Based on the excerpts provided, Gandhi believed that:

- The partition of the country was done with the consent of both the Muslim League and the Congress.
- Gandhi initially believed that satyagraha’s struggles were based on non-violence and spiritual strength, but later realized that it was only passive resistance.
- Gandhi believed that violence and murder were not necessary to achieve their goals and that taking the whole country by friendly negotiations and peaceful methods was possible.
- Gandhi admitted his mistake in realizing that their fight was not one of non-violence, which resulted in the current situation of arson and fearlessness towards the British.
- Gandhi recognized the bravery and self-sacrifice of the revolutionaries in reducing the fear of the English but believed that their approach lacked understanding of non-violence.

- Gandhi acknowledged that the Partition of Bengal caused a significant awakening and resistance among the people, and believed that Bengal would eventually be reunited.
- Gandhi believed that if there was partition, the Muslim majority and the Muslim Government in power would be responsible for it.
- Gandhi emphasized the common cultural and linguistic heritage of Hindus and Muslims in Bengal and urged Hindu brethren to forget the past differences in religion.
- Gandhi's statements about driving the English out of the country, marching with thousands of countrymen to the muzzle of English guns, and praising the heroism of those who threw bombs were seen as unwise and could have negative consequences in a non-political meeting. It is also mentioned that the Chairman of the meeting intervened to prevent further violent interruptions.

As is evident, the response of ChatGPT this time was much more comprehensive, and it contained all aspects of Gandhi's views on the Partition of India as well as the Partition of Bengal. This shows that our algorithm succeeded in scoring the most relevant paragraphs higher than other paragraphs.

2 Additional header files

We have included the C++ header file `<cmath>` only for calculating the logarithm to the natural base, which is required in the formula of *IDF*. We could have implemented the same ourselves using binary search, but it would have been in $O(\log n)$ time compared to the $O(1)$ time of the built-in function. Hence we chose to go with the built-in function instead.

3 Use of LLMs other than ChatGPT

We have used **HuggingChat** by **Hugging Face - Hugchat API** and **Generative AI** by **Google - PaLM2 API** other than **ChatGPT** by **OpenAI - GPT-3.5 Turbo**. PaLM2 has a maximum token length of 8196 while ChatGPT has 4097, and PaLM2 is comparable to GPT 4.0 in terms of performance benchmarks.

The same query is sent to both ChatGPT and Google PaLM API and the results of both are then sent to HugChat API which considers all the relevant points, removes the unnecessary points and outputs the answer in a properly structured format.

Further usage details about the 3 APIs have been given in the next section.

4 Prompt Engineering

We are basically using ChatGPT and PaLM to get outputs for the given excerpts and then using HuggingChat to combine the output of both models. We had to figure out how to get the output in the format desired such that unnecessary information and pleasantries that the LLMs usually output too with the output desired. The following are the prompts given to the models so that they give the correct output format and content.

4.1 Prompt used for ChatGPT and PaLM

We have fed the following prompt to ChatGPT and PaLM after feeding the query to ensure that it does not use its existing data to provide answers:

“I want answers strictly based on the excerpts provided. Read each excerpt line by line and tell me the relevant information from the excerpts based on the question that I provided! DO NOT PROVIDE ANY NOTES or your opinions about the answer to the question. I want you to just output the answer.”

4.2 Prompt used for HugChat

We have fed the following prompt to HugChat after feeding the results returned by ChatGPT and PaLM so that it removes all unnecessary points and gives the best answer as output:

“Without using any other knowledge and just based on these answers, construct the most relevant answer. I want you to take the BEST OF THE TWO ANSWERS if possible and NOT JUST SELECT ONE of them as being more relevant than the other. And just output the final answer and NO OTHER REMARKS/COMMENTS/NOTES of your own.”

“Give more priority to technical information and data over sentence structure when comparing points. I want you to output the final answer combining the most relevant content and points from both answers and not their grammar. DO NOT SAY ”Sure, here’s the combined answer based on given parameters”. ”

4.3 Prompt Engineering Parameters

Additionally, we have fed some parameters in our prompt to fine-tune the response of ChatGPT according to our requirements [4]:

- **Temperature** It controls the randomness and creativity of the generated text. It can vary between 0 and 1.
Value used for ChatGPT and PaLM: 0.4
Value used for HugChat: 0.7
- **Top_P** It controls how many of the highest-probability words are selected to be included in the generated text. It also varies between 0 to 1.
Value used for ChatGPT and PaLM: 0.5
Value used for HugChat: 1.0
- **Frequency Penalty** It is inversely proportional to allowing specific terms to repeat multiple times in a prompt. Higher numbers make the output diverse, promoting rarer words, while negative values get a more common, sometimes repetitive, language.
Value used for ChatGPT and PaLM: 0.6
Value used for HugChat: 0.6
- **Presence Penalty** It ensures output remains coherent, prevents it from sounding unnatural and makes it sound more human-like.
Value used for ChatGPT and PaLM: 0.1
Value used for HugChat: 0.1

- **Stop Sequence** It ensures better structure to generated text.

Character used for ChatGPT and PaLM: “\n”

Character used for HugChat: “\n”

5 Optimizations

5.1 Finding Top-k Paragraphs

We have used min-heap to identify the paragraphs with the greatest k scores. The algorithm used is the same as the “Algorithm 2B” discussed under “The Selection Problem” in the class. It is as follows:

- Only k paragraphs are accepted from the vector of paragraphs (para), and a min-heap is built out of them by calling insert k times, ordered by the respective score of paragraphs (stored in the variable para_score).
- The scores of the rest $(n - k)$ paragraphs are compared one by one with the score of the paragraph stored in the root. (Note: n is the total number of paragraphs)
- If the score of any paragraph is larger, then the root is replaced by that paragraph and *heapifyDown* is called at index 0.
- At the end, the paragraphs stored in the heap are the top-k paragraphs. We have sorted them using the heap-sort algorithm to order them in ascending order of scores.
- Time complexity of the algorithm is: $O(k \cdot \log k) + O((n - k) \cdot \log k) + O(k \cdot \log k) = O(n \cdot \log k)$, which is much better than the complexity of the naive algorithm ($O(n \log n)$).

5.2 Optimal Value of k in Top-k

The value of k in top-k paragraphs should be 5 to get better search results. We tried to use a higher value of k (we toggled the value of k between 4 to 10). The higher k we use, the more probable it is that token limit of ChatGPT will be reached. Google Palm 2 is better than ChatGPT in this matter as it has twice the token length but since we are using both APIs we are still constrained by the ChatGPT token limit. Also, the paragraphs obtained started becoming irrelevant as per our observations for values of k higher than 5 or 6. Feeding only relevant paragraphs to the LLM ensures that the search results are more accurate and less ambiguous.

5.3 Refinement of query

To get better search results, we have ensured that only important keywords in a query contribute to the score of paragraphs. We have ignored common words like “the”, “is”, “what”, etc. We have achieved this by neglecting the contribution of words which have a very high frequency in the general corpus. The list of removed words, stored as a vector in our code is as follows:

Removed words: {“the”, “of”, “and”, “to”, “a”, “in”, “for”, “is”, “on”, “that”, “by”, “this”, “with”, “i”, “you”, “it”, “not”, “or”, “be”, “are”, “from”, “at”, “as”, “your”, “all”}.

These words appear so frequently that even though the term saturation in BM25 is taking care of that, their score is still so high that their relevancy is equivalent to the terms appearing in the query (rarely in the corpus) which skews the results poorly.

References

- [1] Wikipedia, “Okapi bm25,” https://en.wikipedia.org/wiki/Okapi_BM25.
- [2] O. Meriwani, “Smoothing in natural language processing,” 2020, <https://omarmeriwani.com/text-analytics/smoothing-in-natural-language-processing/#:~:text=Laplace%20smoothing%20or%20additive%20smoothing,in%20the%20dataset%20by%20one>.
- [3] R. Seitz, “Understanding tf-idf and bm-25,” 2020, <https://kmwllc.com/index.php/2020/03/20/understanding-tf-idf-and-bm-25/>.
- [4] R. Morais, “Parameters: The secret power to have more control of chatgpt,” 2023, <https://medium.com/@robmoraisjr/parameters-the-secret-power-to-have-more-control-of-chatgpt-d84b9af8e852>.