Exploring Argument Retrieval With Transformers

Aditya Hegde
201IT105
Information Technology
NIT Karnataka
Surathkal, India

Vishruth M
201IT167
Information Technology
NIT Karnataka
Surathkal, India

Buddha Teja
201IT115
Information Technology
NIT Karnataka
Surathkal, India

Samyak Jain
201IT125
Information Technology
NIT Karnataka
Surathkal, India

***Abstract*—This work looks in to the problems faced due to the collection of huge amounts of data in the current digital era which makes it difficult to extract the exact argument(also highlighting the perspective of the document) required from the large corpus of documents, through query optimization and query expansion to get better matches for the required argument in the dataset . This work seeks to enhance the precision of required information retrieval by leveraging the power of transformers in the field of NLP(Natural Language Processing). Specifically, it aims to expand user queries using the transformer decodings and encodings techniques like BERT(Bidirectional Encoder Representations from Transformers), GPT-3(Generative Pre Trained Model) and USE embedder(Universal Sentence Embedding) to generate contextually relevant arguments which further helps in improving document retrievals efficiency and effectiveness.**

Keywords : Transformers, Argument Retrieval, BERT(Bidirectional Encoder Representations from Transformers), GPT-3(Generative Pre Trained Model), USE(Universal Sentence Encoder)

I. INTRODUCTION

Searching for the information in the digital world, be it through search engines web crawlers, through server API end points or by any other method has become vital in our daily lives as we have become more accustomed with the digital world . There are many methods for retrieving information but as there is day to day increase in the amount of data collected in the world, the effective and efficient retrieval of the documents becomes challenging in terms of computational power and time consumption .

Retrieving documents from the currently available massive volumes of data now is an extremely difficult operation. Search engines and information retrieval systems now play a vital role in rapidly and effectively responding to consumers' inquiries due to the exponential rise of diverse web services and digital material. The meaning gaps between the user's query and the documents returned, which frequently lead to lost pertinent information and ineffective searches, are one of the enduring problems with these systems.

The effectiveness of information retrieval methods depends on the precision with which user queries are matched to relevant documents. However, the direct mapping of queries to documents with respect to metrics like term frequency, document frequencies,etc.. faces challenges from the inherent complexity of human language, the variety or different types of user intents, and the vastness of the dataset. Consequently, the need for query optimization and expansion arises as pivotal strategies to enhance the efficiency of information retrieval systems.

Traditional document retrieval methods assume that a single highly-ranked document in the dataset or collection can fully answer the user query. However, this assumption can be misleading when considering queries that inherently involve diverse perspectives or complex relations between the query and documents of the topics. Query expansion becomes crucial to capture the complex or unnoticed information about the user queries, especially when the information sought spans a range of viewpoints or arguments. Our focus on query expansion is driven by the recognition that effective information retrieval should encompass a breadth of opinions, supporting, opposing, and neutral to what the User query is trying to find.

To solve the problem wherein user queries are not enough for the system to match an output or the data in dataset , we plan to implement query expansion through query optimization for it .This becomes essential to bridge the semantic gap between the user's intent and the information in the documents. By refining and expanding on the user quire's, we try to improve the precision and recall of retrieved documents, ensuring that the user is presented with a more relevant set of information.

A. Issues and Challenges

There are many challenges which are faced when we try to produce an optimized and expanded queries. Firstly, the ambiguity in natural language make it challenging to precisely define the user's intent. Users may express the same information need in various ways, leading to mismatches between queries and documents. Additionally, the sheer volume and heterogeneity of dataset increase the difficulty in analyzing the most relevant information.

Focusing on these challenges ,our approach to query expansion for efficient information retrieval is done by leveraging advanced language models like GPT-3, BERT, and USE, we aim to overcome the limitations of traditional retrieval systems and provide a more nuanced, diverse, and contextually relevant set of documents in response to user queries. Through this methodology, we try to enhance the overall search experience and better meet the needs of users in diverse contexts.

B. Motivations for the work

The main motivation behind our work is to provide users with improved access to arguments and evidence within texts, enabling them to make informed decisions, engage in meaningful discussions and extract valuable knowledge from large volumes of information.In this era of information overload and fake news, argument retrieval can be used to fact-check and verify claims made in various texts.In online forums, social media, and other platforms, users often engage in debates and discussions. Argument retrieval can help users locate and reference arguments made by others or find counterarguments to support their own claims.

II. LITERATURE REVIEW

In this literature survey, we look into the existing body of work related to Query expansion and query optimization. Our objective is to provide an in-depth overview of the methodologies, challenges, and advancements in this field, shedding light on the state-of-the-art techniques . Query optimization has emerged as a critical area of research within the field of Information retrieval. In today's society, the demand for high-quality results from search engines remains undiminished, even in challenging conditions faced now as discussed above.

In [1], To implement an effective and novel method in query expansion, it is necessary to commence with a base which has deep insight to it, for this we have selected the paper Exploring Argument Retrieval with Transformers by Christopher Akiki and Martin Potthast published in 2020 . This paper focuses on query expansion methods for search engines. Three query expansion techniques namely, GPT decoding(pre-trained model), BERT encoding (using Masked Learning Model)and USE(Universal Sentence Encoder) encoding were used for the argument retrieval for the modified query, after employing pre-processing techniques for the efficient document retrieval.

while [2] mainly focused on how various features which we might be able to integrate with our proposed methodology for a more effecient system of information retrieval we looked in to the below papers , Argument Retrieval for Comparative Questions based on independent features by Thi Kim Hanh Luu , Jan Niklas Weder published in 2021 . They proposed a pipeline wherin queries are expanded by different methods and the outputs are then merged back the expanded and the original query to then continue the information retrieval process.

The paper [3] Creating an Argument Search Engine for Online Debates by Maximilian Bundesmann , Lukas Christ , and Matthias Richter published in 2020 . They proposed a method through which the retrieval process extend the Lucene search core with an implementation of the DPH concept. This in turn will improve the overall accuracy compare to the baseline model for information retrieval.

The paper [4] Argument Retrieval for Comparative Questions by Ekaterina Shirshakova, Ahmad Wattar published in 2021 proposed a method wherin they used lemmatized document and lemmatized title as search fields along with the weighted query expansion with synonyms to get best results. Initial set of document candidates are re-ranked using elasticsearch to get improved and better-ranked retrieved results. Experimented with different combinations of search fields and expansions, and evaluated them with metrics like ndcg10, recall10, precision and F1-score.

A. Outcome of Literature Review

Through our literature review we were able to come to the conclusions that Query expansion has been extremely beneficial to improve the accuracy and relevancy of results in argument retrieval systems for the respective queries.The current transformer models are being used intensively in new works due to their efficiency in text analysis for these types of tasks. Using Ranking algorithms in selecting the best query applicable to get the most relevant results may take more time but improves accuracy and relevancy of the results.Many query expansion techniques though introduced were not generalized in nature, and if generalized were not as accurate.Natural Language Processing of the corpus greatly helps in improvement of the quality of results obtained and also improving the effectiveness of the expansion of the query as they can also capture hidden meanings.

B. PROBLEM STATEMENT

Improving the efficiency and effectiveness of the document retrievability for a system by enhancing the Argument Retrievability of a User Query using transformer query expansion techniques like GPT decoder, BERT encodings and USE encodings to generate contextually relevant arguments.

III. DATASET

The dataset used for this work is the args.me corpus , it comprises 382,545 arguments crawled from four debate portals in the middle of 2019. The debate portals are Debatewise, IDebate.org, Debatepedia, and Debate.org. The arguments are extracted using heuristics that are designed for each debate portal.This debate dataset has four key attributes argument , conclusion, stance and id . The conclusion provided is the main topic on which a particular debate is conducted and we have the arguments for that particular topic or claim which are either supporting or opposing the claim(indicated by the stance of the argument with respect to the query).

argument string	conclusion string	stance string	id string
The resolution used by Pro *assumes* that Australia isn't already a 'significant' country - however, I…	Australia should be a more significant country	CON	4d3d4471-2019-04-18T11:45:01Z-00000-000
First of all we invented amazing things like WiFi, Google Maps, Polymer bank notes (if you are…	Australia should be a more significant country	PRO	4d3d4471-2019-04-18T11:45:01Z-00001-000
I accept.	Australia should be a more significant country	CON	4d3d4471-2019-04-18T11:45:01Z-00002-000
Australia has always been put into the shadow behind countries like America, Canada and even…	Australia should be a more significant country	PRO	4d3d4471-2019-04-18T11:45:01Z-00003-000
Alright then.	The closet dementia of the superior ego god complex, The bible and why…	CON	fbe6ad2-2019-04-18T11:12:36Z-00000-000
Why is it that so-called christians, Because there is no such a thing as a christian, Have serious…	The closet dementia of the superior ego god complex, The bible and why…	PRO	fbe6ad2-2019-04-18T11:12:36Z-00001-000

Fig. 1. Dataset

```
from collections import Counter

unigram_vocab = Counter()
bigram_vocab = Counter()
print(dataset)

for document in dataset:
    print(document)
    tokens = preprocess(document)  # the preprocessing function
    unigram_vocab.update(tokens)
    bigram_vocab.update(zip(tokens, tokens[1:]))
```

Fig. 2. Code for Unigram and Bigram

IV. WORK DONE

A. EXPERIMENTS DONE AS PART OF LAB ASSIGNMENTS

LAB ASSIGNMENT - 1 :-

1) Subteam 1 Task: We have explored different ways on how to efficiently store and retrieve documents based on the terms in them for effective and efficient query processing. Data preprocessing is very important as it will transform raw data into the format required for analysis or modelling. We have selected suitable preprocessing techniques for the dataset. Pre-Processing Techniques Used :-

1) Word Tokenization :-

Word Tokenization is a fundamental step in Natural Language Processing. It is a process of splitting a text or sentence into individual tokens. Tokens are basic units of text and can be characters or words.

2) StopWord Removal :-

This step involves eliminating words that are considered of little value in texts. These words are mostly frequently occurring words which are of little importance.

3) Stemming :-

Using this text normalization technique we have reduced the words to a much simpler form. It is also observed that the words that are generated are not always meaningful

4) Lemmatization :-

We have implemented Lemmatiztion in order to reduce complex words to their simple form which will help in analysis. It helps in normalizing words by reducing them to their simple form. For example, the words ”running,” ”ran,” and ”runs” would all be lemmatized to the base form ”run”. It is seen that Lemmatization generates only meaningful words

After Pre-processing the document corpus, we designed the BoW model for document corpus using unigram and bigram representations.

Then with the unigram and bigram vocabularies, we have vectorized each document in the corpus using the word or word pair counts.We then calculated and compared the sparsity of the resulting BoW matrices for unigram and bigram representations.For bigram representations, we’re capturing pairs of consecutive words, which might provide some spatial context compared to unigrams. To analyze this, we have calculated the average distance between bigram words within the same document.

2) Subteam 2 Tasks:: We have implemented the Term Frequency-Inverse Document Frequency representation for our document corpus using the vocabulary generated by Sub-team 1 and experimented with two types of data structures namely Hash Map and B-Tree for the better storage and retrieval of the documents with respect to the terms of the document corpus. using hash map data structure, we are storing the document ids in a list that are associated with a particular term in the corpus for the storage and retrieval in terms of key-value pairs, where each key is unique. This Hash Map data structure for the storage and retrieve processes have helped quite a bit as we can retrieve the doc ids in constant time , We have also used a B-tree for the same, B-tree is a balanced tree data structure that maintains sorted data and allows searches, insertions, deletions, and sequential access in logarithmic time. B-tree was mainly helpful in tasks where there are ranged queries during which we need not search the term for every iteration but only at the beginning and B-tree was also helpful when we had regorous insertions and deletions from the corpus as indexing of it takes quite some time for the processing.
A detailed analysis for each data structure’s cost w.r.t storage and retrieval/insertion/update/deletion perspectives.

- Storage: The amount of space taken by both the data structures is O(no. of unique words)
- Retrieval: The amount of time required for retrieving data from Hash Map is of O(constant) as it only depends on the hash function whereas that of B-Tree is O(log(N)) as it depends on the height of the tree.
- Insertion: Hash Map takes O(constant) time for insertion of data as it only depends on the hash function while the amount of time required for inserting data to BTree is O(log(N)) as it is dependent on the height of the tree.
- Update and Deletion: Since it takes O(constant) and O(log(N)) times to retrieve data from a Hash Map and B-Tree respectively, update and deletion also takes the same time complexities as we first need to search for the token to perform the operations.

The next step was to use the constructed inverted index and perform sample Boolean queries of the pattern shown below.

1) term1 AND term2 OR term3
2) term1 OR term2 AND NOT term3

Best time/space complexity that can be achieved assuming that there are N postings lists in the inverted index depends on the specifics of the inverted index implementation, the

nature of your data, the lengths of the posting lists and the query structure.

Time complexity: O(M + N + K), where M, N and K are the lengths of the posting lists for term1, term2 and term3 respectively. Retrieving the posting lists is O(1) on average.

Space Complexity: O(1) because the result posting list doesn't take additional space in the inverted index.

There are different query optimisation techniques to reduce time taken to execute a boolean query.

Query Optimization Strategies:

For Query 1 (term1 AND term2 OR term3):

- Early Termination: If you encounter a document in the term1 AND term2 result that is also in the term3 list, you can stop processing, as it won't affect the final result.
- Skip Pointers: We can use skip pointers within the posting lists to quickly skip over large sections of the list if they don't contain matching documents.

For Query 2 (term1 OR term2 AND NOT term3):

- Early Termination: As in the previous query, if you find a document in the term2 list that is also in the term3 list, you can exclude it without processing the entire term1 list.
- Caching: We can cache the result of term2 AND NOT term3 to avoid re-computation if the same query is requested multiple times.

V. METHODOLOGY

Since the very beginning of the digital era, it's been very popular requirement and a challenge to retrieve the related documents for the query that has been raised by the User. With the rapid increase in the amounts of data, it's has also been a challenge to retrieve those documents which are arguing on the topic which the User has searched, as those set of documents are also providing a perspective on the query, mostly either supporting or opposing the user opinions further helping understanding of the User. Thus Argument Retrieval for the query is performed to retrieve those document which also holds a perspective on the User Query.

Argument Retrieval is a paradigm in which we do not just retrieve the documents or information from the document corpus but we also extract that information which discusses a particular perspective of the claim and argues against or supporting the raised query.

In this paper, we have employed query expansion techniques using transformers for retrieving the arguments. We firstly expanded the query as it is the phase where we can mine the query and add any information that improves the semantic relation with the document as in general the query only contains the crux of the requirements. During the query expansion, we have tried adding various new words and phrases which reflect the same meaning or are synonymous with the considered query, or any polysemy's(words with different meaning) possible.Thus further adding the meaning that is undiscovered to the query.

We have employed two major techniques for query expansion that are based on transformers and a sentence embedder in pipeline ,

(i) BERT (Bi-directional Encoder Representation from transformers) architecture: BERT is a traditional transformer model architecture which has been trained on large corpus of data and thus is useful in capturing the context of the texts especially because of the attention mechansim of the transformers along with bidirectional encoding property of the architecture, capturing the context of the word and also importance of the word from both the sides of the text further helping in deciding the weight of the word and can also be used for tasks like word prediction, by making the model predict a [MASK] in the text.

As BERT can be used as Masked Language Model(MLM) instead of auto regressively generating the text, we have used BERT in the word prediction process, we have generated words (5 for every term token) for every term(after data cleaning) in the query marking it as mask thus getting to know the details about context as the words explained more about the query. Thus after generating words for a term in the query we added all the terms along with the initial masked term into a single string which gives us the expanded version of the query.

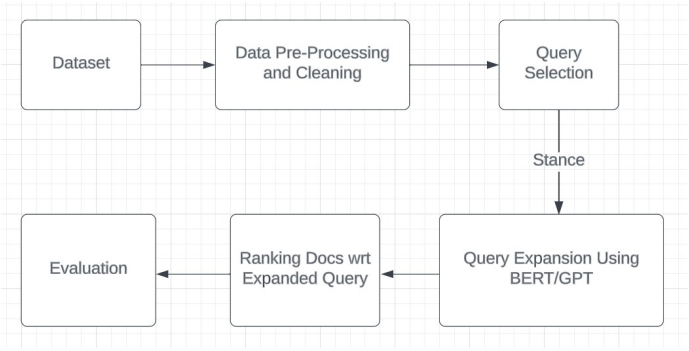

Fig. 3. Flowchart of BERT

(ii) GPT-3 (Generative Pre-Trained Model) architecture: GPT models are pretrained on a very large corpus of data, which helps in recognizing and capturing the text patterns. It is used for autoregressively generating text for the given query based on the context of the prompt, GPT models also finds it's use cases in the field of query expansion as we can feed the query along with stance as a prompt and expand the prompt retaining the context and also finding any unnoticed meanings.

We have used a GPT-3 model to expand the query that is considered. We have not trained the data on the dataset as we are trying to generate a pipeline that can be used on any data source rather than a particular scenario or case specific dataset. we have also generated the prompts based on the stance for an argument i.e; for a considered query we have derived prompts that are basically supporting and opposing the query and then feed it to the GPT model for

query expansion. After the query is expanded we have made a dataset which only contains the expanded query and the list of documents that are associated with that particular (query + stance) ans similarly we have noted down the expanded queries manually and evaluated it's performance accordingly. Here, user can ask for a specific stance related argument or by default option is considered, in which all the stances for the query are evaluated.

(iii) USE (Universal Sentence Encoder):

USE is a pre-trained model which is trained for various tasks related to the context of the text, In here the text is embedded into a 512 dimensional vector space eliminating the highly computational Query Expansion part and can also be used for varying text sizes.

Though USE have similar architecture as that of BERT, our dataset which has high document sizes can avail length flexibility provided by USE and we have used USE as a sentence embedder to fit both the documents and the queries into a 512 coordinate vector space and then similarity between the query and documents via the distance measure L2 distance using which the documents are ranked accordingly(The more the distance the less the), thus the K nearest neighbours are selected

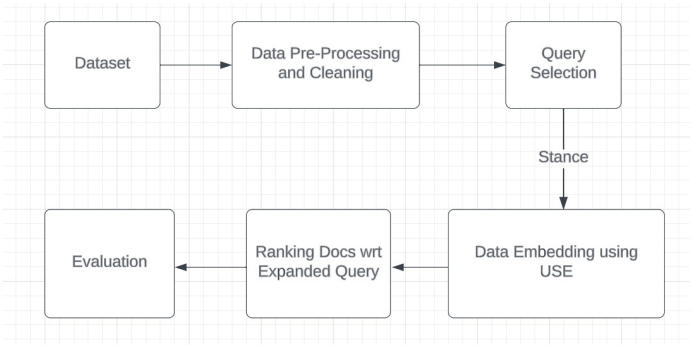

Fig. 4. Flowchart of USE

VI. EVALUATION

After the query expansion based on stance, we have compared the expanded queries with the documents and obtained the similarity measure using the cosine_similarity between each of the query and document pairs. However as the test results for the data is not given, thus we have designed our own evaluation metric, in which we have ranked the documents and queries based on the similarity measures and then we have set a threshold of 3 and 5 lists of rankings of the documents(We have made sure that every query that is considered from the dataset has atleast two positive and two negative stances) for a given query and if the list of rankings contain the document that is related to we mark the query expansion as useful.We have also computed weighted rankings of the similarities in the performance evaluation instead of plain ranking summation

VII. RESULTS AND ANALYSIS

We have pre-processed the dataset using various data cleaning and visualization techniques. We have implemented query expansion algorithms along with the stance using transformer models like BERT(Bidirectional Encoder Representation from Tranformers),USE(Universal Sentence Encoder) and GPT(Generative Pre Trianed Model) for the Argument retrieval.

We have also developed an evaluation criterea for the argument retrieval which have helped us evaluate the performances of the query expansion algorithms as,

For the BERT model we have verified for Query sample sizes of 250,500 and 1000 and their related documents It is observed that for a ranking list of size 3 the accuracy is around 55.5 precentage and for ranking list of size 5 the accuracy is around 64.3 percentage.

Queries	% in Top 3	% in Top 5
250	59.37%	70.31%
500	53.62%	62.32%
1000	53.40%	60.22%

Fig. 5. BERT Accuracy

For the USE model we have worked on around 500 queries and the accuracy for top 3 list rankings and top 5 list rankings are around 70 and 76 percentages

Queries	% in Top 3	% in Top 5
250	76.56%	82.80%
500	67.39%	73.18%
1000	66.11%	72.08%

Fig. 6. USE Accuracy

We have also worked on manually computing the expanded query for the modified prompt using chatGPT for nearly 20Queries and it is observed that the efficiency for that is also considerable better

VIII. CONCLUSION AND FUTURE WORK

We have worked on different query expansion and Sentence embedding algorithms and evaluated the real-time argument retrieval performance for an online debate dataset.

We have experimented with algorithms on how exactly they impact the documents based on their parametrics like the size, number of quries related to them.

We could see that for our dataset the USE(Universal Sentence Encoder) model significantly outperformed the other models like BERT(Bi-directional Encoder Representation from Transformers)

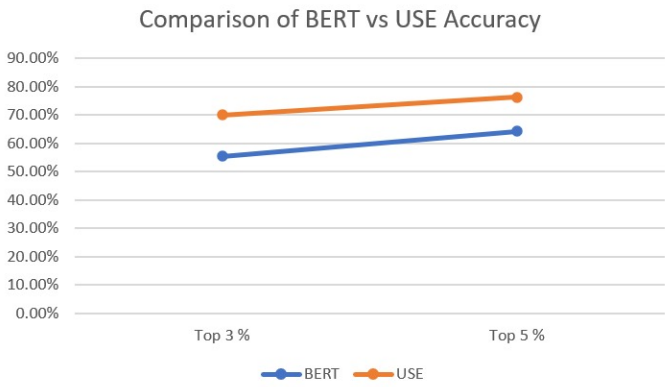

Fig. 7. Results

REFERENCES

[1] Christopher Akiki and Martin Potthast, "Exploring Argument Retrieval with Transformers", Notebook for the Touché Lab on Argument Retrieval at CLEF 2020.

[2] Ekaterina Shirshakova, Ahmad Wattar , "Thor at Touché 2021: Argument Retrieval for Comparative Questions" Notebook for the Touché Lab on Argument Retrieval at CLEF 2021

[3] Thi Kim Hanh Luu , Jan-Niklas Weder, "Argument Retrieval for Comparative Questions based on independent features" Notebook for the Touché Lab on Argument Retrieval at CLEF 2021

[4] Maximilian Bundesmann , Lukas Christ , and Matthias Richter , "Creating an Argument Search Engine for Online Debates" Notebook for the Touché Lab on Argument Retrieval at CLEF 2020.

[5] Staudte, C., & Lange, L. (2020). SentArg: A Hybrid Doc2Vec/DPH Model with Sentiment Analysis Refinement. Conference and Labs of the Evaluation Forum.

[6] Akiki, C., Fröbe, M., Hagen, M., & Potthast, M. (2021). Learning to Rank Arguments with Feature Selection. Conference and Labs of the Evaluation Forum.

[7] Green, T., Moroldo, L., & Valente, A. (2021). Exploring BERT Synonyms and Quality Prediction for Argument Retrieval. Conference and Labs of the Evaluation Forum.

[8] Dumani, L., & Schenkel, R. (2020). Ranking Arguments by Combining Claim Similarity and Argument Quality Dimensions. Conference and Labs of the Evaluation Forum.

[9] Entezari, S., & Völske, M. (2020). Argument Retrieval Using Deep Neural Ranking Models. Conference and Labs of the Evaluation Forum.

[10] Raimondi, E., Alessio, M., & Levorato, N. (2021). A Search Engine System for Touché Argument Retrieval task to answer Controversial Questions. Conference and Labs of the Evaluation Forum.

[11] Alecci, M., Baldo, T., Martinelli, L., & Ziroldo, E. (2021). Development of an IR System for Argument Search. Conference and Labs of the Evaluation Forum.

IX. APPENDIX

LAB-ASSIGNMENT-2 :-

1) Subteam 1 Task
   a) Storing Documents and Queries:
      - We have loaded documents then after tokenizing them into words, and stored them in a list called 'documents'.
      - Queries are read from a file, tokenized, and stored in a list named 'queries'.
   b) Preprocessing:
      - The documents and queries are preprocessed by converting them to lowercase and tokenizing them using the NLTK library.
      - Stop words (commonly occurring words like 'the', 'is', 'at', etc.) are removed from the tokenized documents.
   c) Calculating Document Frequencies:
      - Document frequencies for each term in the queries are calculated to determine how many documents contain each term.
2) Subteam 2 Task
   a) BM11, BM15, and BM25 Calculation:
      - For each query-document pair, BM11, BM15, and BM25 scores are calculated.
      - These scoring methods are variations of the Okapi BM25 algorithm, a ranking function used by search engines to rank matching documents according to their relevance to a given search query.
      - Scores are computed based on term frequency (tf), document frequency (n), average document length, and other parameters specific to each BM scoring method.
   b) Ranking the Results:
      - The calculated scores for each query-document pair using BM11, BM15, and BM25 are stored in separate lists ('bm11results', 'bm15results', 'bm25results').
      - These lists are sorted to rank documents in descending order based on their scores for each query.
   c) Displaying Top-ranked Documents:
      - For each query, the code prints the query, followed by the top-ranked documents along with their scores, separately for BM11, BM15, and BM25.
   d) Output:
      - The output displays the query number followed by the ranked documents for each query based on BM11, BM15, and BM25 scoring methods.