IRE Assignment 1 Report

Name - Pranav Gupta

Roll no. - 2021101095

Task 1: Preprocessing and Tokenisation

In this task, the text is prepared for retrieval. Punctuation, special characters, and numbers
are removed using the regular expression [ˆa-zA-Z]. This makes the text contain only
alphabetic letters and spaces. Then, the text is separated into tokens, which are essentially words, by splitting the text on the basis of a space. This process is done in order to obtain raw data.

Following this, Porter Stemming Algorithm is used to reduce words to their base or root form. It is a very popular method in NLP to reduce words to their root form. It applies heuristic rules to remove common suffixes from words, simplifying them for analysis. For example, ″playing″ becomes ″play″ after stemming. In this task, the tokens were stemmed using the PorterStemmer from nltk.stem. This gives the root forms of the tokens.

Task 2: Computation of tf-idf and BM25

In this task, the Term Frequency (tf) and TF-IDF (tf-idf) are computed for each document.
Subsequently, the top p stems in each document are compared based on both tf and
tf-idf. These two metrics are essential in information retrieval and text mining, aiding in
understanding the importance of terms within a document and across a collection.

Term Frequency (tf)

tf measures how often a given term appears in a specific document. It is

calculated as:
tf(ti, dj) = freq(ti, dj)/summation over all k(freq(ti, dk)) where, tf(ti, dj) is the term frequency of term t in document d and freq(ti,dj) is the number of occurrences of the term ti in document dj. The purpose of tf is to normalize the term frequency by dividing it by the length of the document (i.e. the number of terms in the document). This normalization removes biasness from longer documents so that longer documents don't have a natural advantage in the sense that they will have more occurrences of different terms in the corpus.

Inverse Document Frequency (idf)
idf measures the importance of a term in the entire corpus. It is calculated as:
idf(ti) = log(N/nti) where, idf(t) is the Inverse Document Frequency of term ti, N is the total number
of documents in the corpus and nti is the number of documents containing term ti. idf gives higher weight to terms that are rare across the entire corpus and lower weight to common terms because rarer words add more value to the meaning and representation of the corpus.

Term Frequency-Inverse Document Frequency (tf-idf)
tf-idf combines tf and idf to assess the importance of a term within a document and the
entire corpus. It is calculated as: tf-idf(t, d) = tf(t, d) × idf(t)
The tf-idf value reflects how significant a term is within a document (tf) while considering
its rarity across the entire corpus (idf). High tf-idf values are obtained for terms that are
frequent within a document but relatively uncommon across the corpus.

BM25

BM25 is a ranking function used by search engines to estimate the relevance of documents to a given search query. It improves upon the classic TF-IDF model by incorporating document length normalization and tunable parameters. It's major strength lies in the fact that it calculates a similarity score of a document given a query. That is, it takes into account the words appearing in the query and then normalises/penalises the larger documents.

The BM25 score for a document *D* and query *Q* is calculated as:

$Score(D, Q) = \text{sum over all } i \ IDF(q_i)*f(q_i, D)*(k_1+1)/(f(q_i, D) + k_1*(1 - b + b*|D|/avgdl)$

where q1, q2, ... qn are terms appearing in the query, $f(q_i, D)$ is the term frequency of q_i term in document D, $|D|$ is the length of document D, avgdl is the average document length in the collection, k1 and b are hyperparameters where k1 controls the impact of term frequency and b controls the impact of document length.

IDF(qi) is calculated similarly to TF-IDF: $IDF(q_i) = \log (1+(N - n(q_i) + 0.5)/(n(q_i) + 0.5))$ where N is the total number of documents in the corpus and (q_i) is the number of documents containing q_i.

Following the implementation using these formulas, experimentation with different values of b was done in order to observe the impact of b on the results.

Implementation of Retrieval Models

Boolean Model

In the Boolean Model, a Term-Document Matrix class is implemented with the goal of
creating Boolean models based on the top p stems extracted from a corpus of text documents.
The key components and steps involved in this process include:

• Boolean Matrix Creation: A Boolean matrix is constructed where each row corresponds to a unique term, and each column represents a document. In the Boolean matrix, '1' indicates the presence of a term in a document, while '0' signifies
absence.

• Boolean Query Representation: The user's input query undergoes a series of transformations to create a representation suitable for Boolean querying. Firstly, the
query is processed following the same steps as performed above for document processing which leads to the formation of tokens of the query as well. This step aligns the query terms with the stemmed terms present in the term-document

matrix, significantly enhancing the likelihood of precise matching. The final step in query representation involves the creation of Boolean Vectors. In these Boolean Vectors, each position corresponds to a document in the corpus. A '1' in a particular position of the Boolean Vector indicates the presence of the query term in the corresponding document, while '0' signifies its absence. For example, consider the query "Hi, this is Pranav." This query is converted into a structured representation: [[Boolean Vector representing 'Hi'], [Boolean Vector representing 'this'], [Boolean Vector representing 'is'], [Boolean Vector representing 'Pranav']]. This representation allows for precise and efficient logical comparisons with the term-document matrix, ultimately facilitating accurate document retrieval based on the user's Boolean query expressions.

Based upon the value of p (Most occurring p stems in the corpus), only those terms are searched in the documents as well as the queries, i.e Vectors of length p are formed for both the documents as well as queries where a value of 1 indicates presence of that particular word while 0 indicates it's absence. Then, following this, dot product of each document vector is taken with each query and only those documents that give a vector of all 1s on dot product with the query vector are retrieved, other documents are simply discarded. In this manner, Documents are retrieved for each query.

In this model, p is a hyperparameter because the number of stems to be considered highly influences the retrieval process. More commonly occuring stems such as stopwords are present in almost all the documents and this is the reason why in case of lower values of p (1-2) without stopwords removal, almost all documents in the corpus are retrieved since they are present in almost all the documents. If we want to capture meaningful information, it is only obtained either by increasing values of p, thus penalising the model to have to contain the p stems under consideration, which leads to some context capture, and to some extent, meaningful information retrieval. The other solution is to simply remove stopwords and then apply the Boolean retrieval Model. So, in order to perform good retrieval, we need to consider intermediate values of p and Stopwords Removal as well otherwise model will retrieve the documents including the stopwords themselves, which (stopwords) generally don't convey much information.

```
{'the': 1839673}
For Query 0, Value of p = 1, the Documents retrieved are:  7580
For Query 1, Value of p = 1, the Documents retrieved are:  7580
For Query 2, Value of p = 1, the Documents retrieved are:  7580
For Query 3, Value of p = 1, the Documents retrieved are:  7580
For Query 4, Value of p = 1, the Documents retrieved are:  7580
For Query 5, Value of p = 1, the Documents retrieved are:  0
For Query 6, Value of p = 1, the Documents retrieved are:  7580
For Query 7, Value of p = 1, the Documents retrieved are:  7580
For Query 8, Value of p = 1, the Documents retrieved are:  7580
For Query 9, Value of p = 1, the Documents retrieved are:  7580
```

```
{'the': 1839673, 'of': 751071}
For Query 0, Value of p = 2, the Documents retrieved are:  0
For Query 1, Value of p = 2, the Documents retrieved are:  0
For Query 2, Value of p = 2, the Documents retrieved are:  0
For Query 3, Value of p = 2, the Documents retrieved are:  0
For Query 4, Value of p = 2, the Documents retrieved are:  0
For Query 5, Value of p = 2, the Documents retrieved are:  0
For Query 6, Value of p = 2, the Documents retrieved are:  7577
For Query 7, Value of p = 2, the Documents retrieved are:  7577
For Query 8, Value of p = 2, the Documents retrieved are:  0
For Query 9, Value of p = 2, the Documents retrieved are:  7577
```

```
{'the': 1839673, 'of': 751071, 'to': 746043}
For Query 0, Value of p = 3, the Documents retrieved are:  0
For Query 1, Value of p = 3, the Documents retrieved are:  0
For Query 2, Value of p = 3, the Documents retrieved are:  0
For Query 3, Value of p = 3, the Documents retrieved are:  0
For Query 4, Value of p = 3, the Documents retrieved are:  0
For Query 5, Value of p = 3, the Documents retrieved are:  0
For Query 6, Value of p = 3, the Documents retrieved are:  7571
For Query 7, Value of p = 3, the Documents retrieved are:  0
For Query 8, Value of p = 3, the Documents retrieved are:  0
For Query 9, Value of p = 3, the Documents retrieved are:  0
```

```
{'the': 1839673, 'of': 751071, 'to': 746043, 'in': 670002}
For Query 0, Value of p = 4, the Documents retrieved are:  0
For Query 1, Value of p = 4, the Documents retrieved are:  0
For Query 2, Value of p = 4, the Documents retrieved are:  0
For Query 3, Value of p = 4, the Documents retrieved are:  0
For Query 4, Value of p = 4, the Documents retrieved are:  0
For Query 5, Value of p = 4, the Documents retrieved are:  0
For Query 6, Value of p = 4, the Documents retrieved are:  0
For Query 7, Value of p = 4, the Documents retrieved are:  0
For Query 8, Value of p = 4, the Documents retrieved are:  0
For Query 9, Value of p = 4, the Documents retrieved are:  0
```

As shown above, no documents are retrieved for a value of p greater than or equal to 4. That is, as the value of p increases, model tries to look for only meaningful documents based upon the occurrence of the p most occurring stems.

This Boolean Model provides a robust framework for document retrieval and information
retrieval tasks, allowing users to effectively query the corpus based on the top p stems. Whether it's searching for specific topics or customizing the number of top stems for relevance ranking, this model offers flexibility and control over the retrieval process. By transforming user queries into Boolean representations and comparing them with the Term Document Matrix, this model empowers users to retrieve precise and relevant documents, enhancing the efficiency of information retrieval tasks.

Vector Model

The key components and steps involved in this process include:

• Vector Matrix Creation: A Vector matrix is constructed where each row corresponds to a unique term, and each column represents a document. In the Vector Matrix, each cell represents the tf – idf value of the term, with respect to the document. These values, denoting term significance within specific documents and across the entire corpus, lend a nuanced and context-aware representation.

• Vector Query Representation: The user's input query undergoes a series of transformations to create a representation suitable for Vector querying. Firstly, all the queries are preprocessed using the same transformations mentioned above which leads to good representation of a query in the vector space. This step aligns the query terms with the stemmed terms present in the term-document matrix, significantly enhancing the likelihood of precise matching.

The next step in query representation involves the creation of a vector. The vector is created with 2 methods: 1)TF-IDF 2) BM25.

1. TF-IDF: The vector is created with a length equal to the number of unique terms in the documents. Each element in this vector is initially set to zero. The Term Frequency (tf) of each term in the query is calculated, along with the Document Frequency (df). Using the df values, idf is calculated for each term and the vector stores the tf-idf value at the corresponding position. This vector can be used for comparing the query to the documents to find the the most relevant ones.

2. BM25: The vector is created with a length equal to the number of queries posed (which is 10 in this case). Each element in this vector is initially set to zero. The BM25 Score of each term in the query is pre-calculated in the BM25 Matrix and is simply extracted from there. Using these values, a vector is created where each dimension in the vector space indicates BM25 similarity score of the document with the query. Then, all the queries are also represented by creating a basis of all the query vectors in the vector spaces. For this, the same steps of preprocessing for queries is also done. In this manner, we have our document and query vectors ready.

• Vector Query Comparison: With the Vector representation of the query and the TDM ready, the similarity between the user's query vector and each document's vector

in the TDM is calculated using cosine similarity. The formula for cosine similarity is given by:

Cosine Similarity = $A \cdot B/(\lVert A \rVert \lVert B \rVert)$

It assesses how closely the query aligns with each document in terms of the terms

they contain and their importance.

• Result: The the top ′n′ documents that exhibit the highest cosine similarity with the user's query are identified. Here the value of n is also a hyperparameter and is fixed at a value of 5 for the Assignment. These documents are considered the most relevant matches to the query. If a document has a similarity score of zero, it is typically excluded from the results, as it does not match the query's content. The document names along with their similarity measures are returned as the final result.

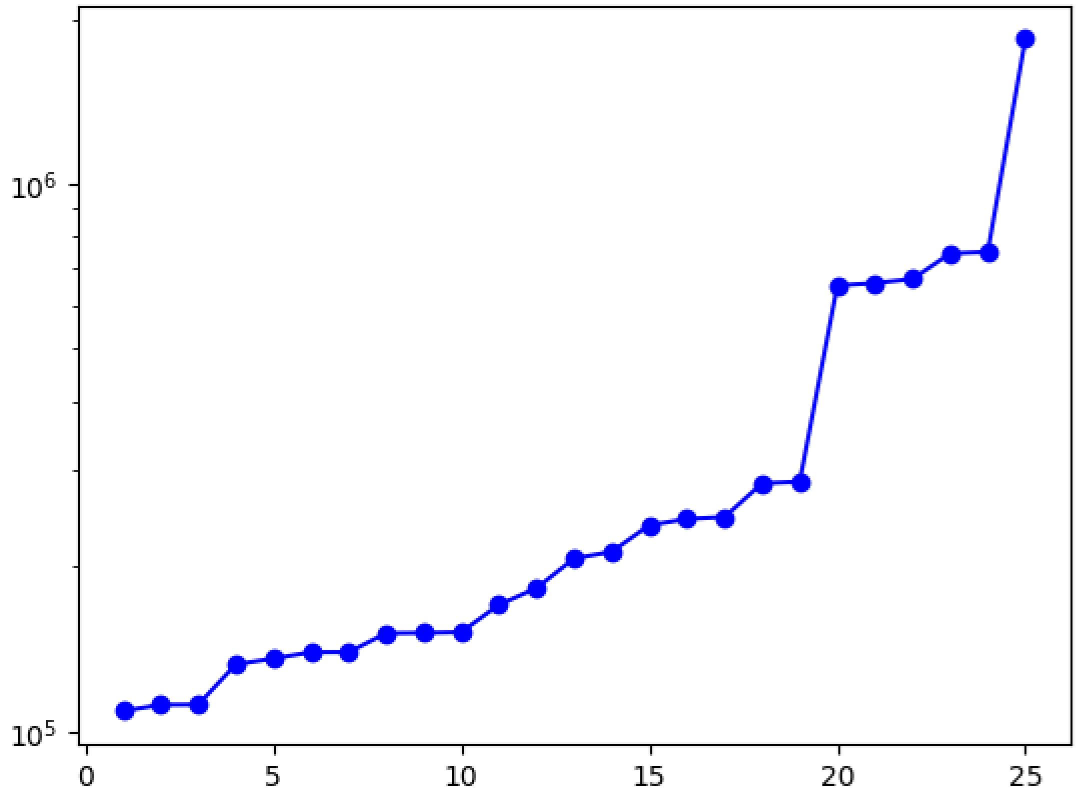
The Vector Model leverages vector representations of the query and documents, along
with cosine similarity, to find and rank the most relevant documents in the corpus that
match the user's information needs. This approach enhances the efficiency and effectiveness
of document retrieval and information retrieval tasks by considering the semantic similarity
between the query and documents.

Task 3: Stopwords removal

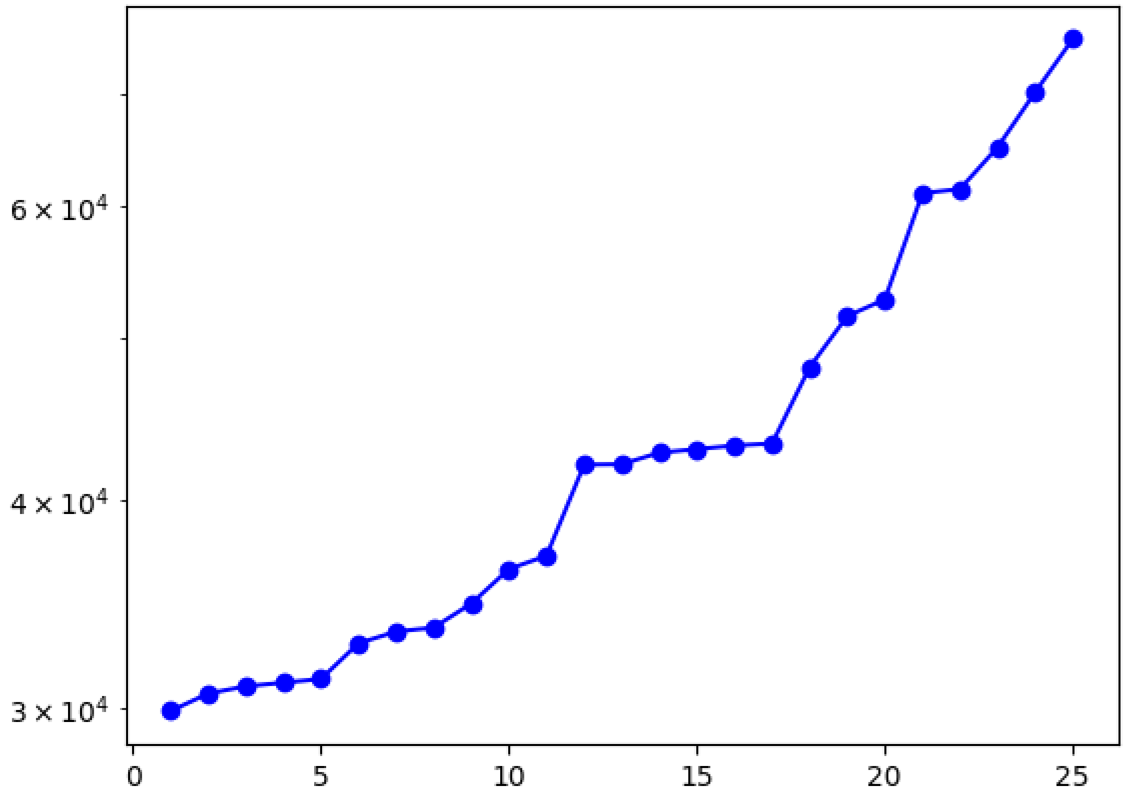
Stopwords are commonly used words in a language (such as "the," "is," "in," "and," etc.) that are often considered unimportant for understanding the main content of a text. They are typically filtered out in natural language processing (NLP) tasks because they don't carry much meaning or contribute to the analysis of the text.

For example, in text mining and search engines, removing stopwords helps focus on the significant words in a document, making text processing more efficient.

The english.stop file containing stopwords of English Language was loaded. Changes were only to the existing class where a check was written if we have to remove stopwords or not. Then, the stopwords were removed from all the documents as well as queries so that they do not interfere with the learning of the model. Following are the graphs for the frequency distributions obtained before and after performing stopwords removal.



Frequency Distribution of words before Stopwords removal



Frequency Distribution of words after Stopwords removal

Top-10 Stems in the Vocabulary before Stopwords Removal and their respective frequencies: {'the': 1839673, 'of': 751071, 'to': 746043, 'in': 670002, 'and': 658631, 'a': 651987, 'on': 285587, 'that': 283587, 'for': 246090}

Top-10 Stems in the Vocabulary after Stopwords Removal and their frequencies: {'peopl': 75593, 'state': 70234, 'year': 65082, 'presid': 61423, 'trump': 61067, 'govern': 52724, 'report': 51523, 'countri': 48006, 'time': 43197, 'polic': 43076}

We notice that now the frequencies of the 25 most occurring words has significantly reduced. It is because the most commonly occurring words are the stopwords themselves and removing them gives an opportunity to the lesser frequency words to appear in the learning of the model. Also, be observing the Top p-stems in the corpus after stopwords removal, words that do not contribute anything to the learning process have been removed which results in better generalisation.

Task 4: Retrieval with Stopwords removal

Following the data preprocessing after stop words removal (i.e. Stemming, frequency counts of terms in documents, etc.), the tf-idf score was computed using the formulas given in Task 2. The obtained tf-idf values are different from those obtained in Task 2. BM25 metric was also calculated for the same text, using the same set of hyperparameters. We notice that BM25 values have drastically reduced compared ti the case when stopwords were not removed.

In the Boolean retrieval model, even for very small values of p(1, 2), the number of documents retrieved have significantly reduced compared to the case when they were not removed. This points us to the fact that stopwords play a very prominent role in case of Boolean Retrieval model.

In contrast, in case of Vector Space Model, which uses tf-idf and BM25 metrics, the documents retrieved are more or less the same. This proves the point that these metrics heavily penalise frequently occurring words which prevents the model from being biased towards particular words, thus painting a better picture in terms of information retrieval. In case of BM25 metric, however, a small difference is obtained due to the fact that BM25 is a stronger metric compared to TF-IDF and based upon mathematical manipulations, it captures the context of the relevant Documents in a better way which leads to better retrieval in the longer run which is evident from the fact that the clustering of documents is better in case of BM25 (results shown below).

```
For Query 0, the Documents retrieved according to TF-IDF Metric are:    6830.txt        5166.txt        1595.txt        5313.txt        192.txt
For Query 1, the Documents retrieved according to TF-IDF Metric are:    1760.txt        287.txt 1244.txt        5322.txt        4203.txt
For Query 2, the Documents retrieved according to TF-IDF Metric are:    4680.txt        3836.txt        2220.txt        6060.txt        2376.txt
For Query 3, the Documents retrieved according to TF-IDF Metric are:    970.txt 6810.txt        1100.txt        6967.txt        1992.txt
For Query 4, the Documents retrieved according to TF-IDF Metric are:    4250.txt        7100.txt        821.txt 2705.txt        1306.txt
For Query 5, the Documents retrieved according to TF-IDF Metric are:    834.txt 1306.txt        982.txt 7490.txt        933.txt
For Query 6, the Documents retrieved according to TF-IDF Metric are:    2310.txt        1369.txt        3430.txt        3300.txt        1539.txt
For Query 7, the Documents retrieved according to TF-IDF Metric are:    2855.txt        1160.txt        6083.txt        6867.txt        4920.txt
For Query 8, the Documents retrieved according to TF-IDF Metric are:    6600.txt        2819.txt        5500.txt        2901.txt        6114.txt
For Query 9, the Documents retrieved according to TF-IDF Metric are:    980.txt 5099.txt        7469.txt        1907.txt        1759.txt
For Query 0, the Documents retrieved according to BM25 Metric are:      4874.txt        5575.txt        5232.txt        1177.txt        3958.txt
For Query 1, the Documents retrieved according to BM25 Metric are:      1760.txt        35.txt  2184.txt        409.txt 3132.txt
For Query 2, the Documents retrieved according to BM25 Metric are:      4827.txt        4806.txt        4136.txt        4678.txt        4406.txt
For Query 3, the Documents retrieved according to BM25 Metric are:      970.txt 1100.txt        3015.txt        1007.txt        7518.txt
For Query 4, the Documents retrieved according to BM25 Metric are:      4250.txt        1716.txt        4953.txt        334.txt 4692.txt
For Query 5, the Documents retrieved according to BM25 Metric are:      4905.txt        6357.txt        849.txt 834.txt 7364.txt
For Query 6, the Documents retrieved according to BM25 Metric are:      568.txt 3300.txt        5687.txt        2310.txt        1610.txt
For Query 7, the Documents retrieved according to BM25 Metric are:      1160.txt        5299.txt        5823.txt        598.txt 4920.txt
For Query 8, the Documents retrieved according to BM25 Metric are:      5291.txt        6600.txt        6795.txt        3127.txt        6114.txt
For Query 9, the Documents retrieved according to BM25 Metric are:      1013.txt        2617.txt        6135.txt        3402.txt        4047.txt
```

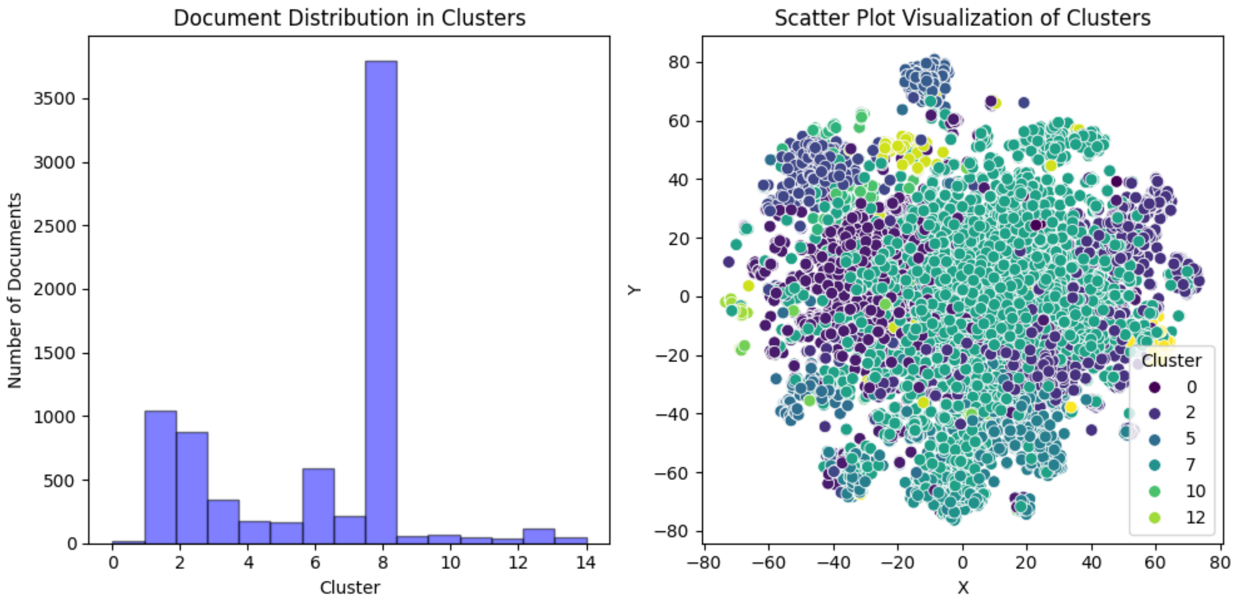
Documents Retrieval without Stopwords Removal

```
For Query 0, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	6830.txt	5166.txt	1595.txt	5313.txt	192.txt
For Query 1, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	1760.txt	287.txt 1244.txt	5322.txt	4203.txt
For Query 2, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	4680.txt	3836.txt	2220.txt	6060.txt	2376.txt
For Query 3, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	970.txt 6810.txt	1100.txt	6967.txt	1992.txt
For Query 4, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	4250.txt	2705.txt	7100.txt	1306.txt	933.txt
For Query 5, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	834.txt 1306.txt	982.txt 7490.txt	933.txt
For Query 6, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	2310.txt	1369.txt	3430.txt	3300.txt	1539.txt
For Query 7, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	2855.txt	1160.txt	6083.txt	6867.txt	4920.txt
For Query 8, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	6600.txt	2819.txt	5500.txt	2901.txt	1365.txt
For Query 9, the Documents retrieved according to TF-IDF Metric after Stopwords Removal are:	980.txt 5099.txt	7469.txt	1907.txt	1886.txt
For Query 0, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	5232.txt	4874.txt	6830.txt	5710.txt	1869.txt
For Query 1, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	694.txt 1760.txt	5322.txt	4894.txt	3132.txt
For Query 2, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	4827.txt	1676.txt	4680.txt	3257.txt	5312.txt
For Query 3, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	5381.txt	970.txt 5479.txt	905.txt 1007.txt
For Query 4, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	4250.txt	785.txt 2557.txt	1716.txt	4191.txt
For Query 5, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	2082.txt	834.txt 2397.txt	3211.txt	247.txt
For Query 6, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	2310.txt	1369.txt	3300.txt	6430.txt	2729.txt
For Query 7, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	1160.txt	598.txt 7299.txt	7002.txt	4920.txt
For Query 8, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	6600.txt	1860.txt	5500.txt	4529.txt	5090.txt
For Query 9, the Documents retrieved according to BM25 Metric after Stopwords Removal are:	980.txt 7470.txt	4047.txt	4974.txt	2362.txt
```

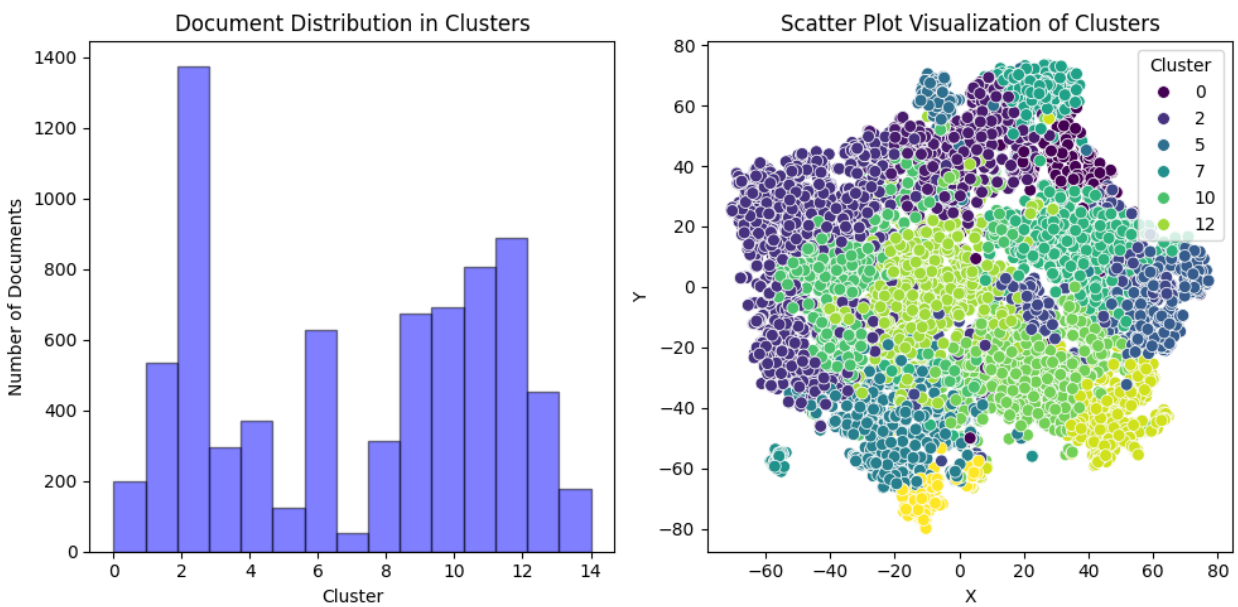
Documents Retrieval with Stopwords Removal

Task 5: Document Clustering

In this part, the tf-idf and the BM25 Matrices created in Task 4(Retrieval with Stopwords removal) are considered since they portray a better picture of information of the document. K-means clustering from scikit-learn was imported, and the above mentioned matrices given as input to the function. This function returns, as output the cluster centers to which a particular document belongs. We observe that in case of tf-idf matrix, majority of the documents are assigned to cluster no. 1, 2 and 8 while in the case of BM25 matrix, documents are uniformly spread out in different clusters, again referring to the fact that BM25 is a stronger version of TF-IDF matrix, and penalises the terms in such a manner that the overall context is better captured. Since the context of all the matrices are almost similar and so, should be assigned uniformly across different clusters, it is not the case with TF-IDF Matrix. So, BM25 metric outperforms tf-idf metric in this case and is generally better than the latter.



Document Clusters obtained in the case of tf-idf matrix



Document Clusters obtained in the case of BM25 matrix

Task 6: Conclusion

The key findings of experiments and analysis and the strengths and weaknesses of each retrieval model (Boolean, Vector, TF-IDF, BM25) are as follows:

The Vector Model, which is constructed using the top stems after removing stop words, in general performs better than Boolean Model for document retrieval tasks. It excels in accurately
categorizing both relevant and non-relevant documents and goes a step further by ranking
them based on their cosine similarity scores. This ranking capability allows users to identify
the most relevant documents with greater precision, enhancing the overall effectiveness of
the retrieval process. One of the limitations of Vector model is that it ignores the order in which these words occur in the corpus, which leads to loss of interesting patterns of data in the corpus.

Internally comparing tf-idf and BM25 metric, we observe that these metrics are not much sensitive to the presence/absence of stopwords in the dataset since they inherently penalise the larger occurring stems in the corpus, allowing for a more robust and better feature extraction model. In case, of Document Clustering, we observe that BM25 leads to formation of more uniform clusters meaning the fact that it is able to capture thecontext of queries and documents and able to better cluster the documents into different genres. This clearly shows that out of the 2, BM25 is superior to tf-idf metric.
On the other hand, the Boolean Model, while also employing the top stems with stop
words filtered out, offers correct results up to a certain extent. However, it falls short in the
crucial aspect of ranking these documents effectively. In this model, most of the relevant
documents receive identical similarity scores, making it challenging to distinguish between
them based solely on relevance. This limitation can potentially hinder the user's ability to
prioritize and access the most pertinent information within the search results.
Also, one of it's other limitations include the fact that it is sensitive to the presence

of stopwords in the corpus, which is not much of a problem for Vector Model. Hyperparameter p also decides the performance of Boolean Model. In general, it is preferred that a higher value of p is used for efficient retrieval.

Impact of stemming and stop word removal on the retrieval performance:

Stemming:

Stemming is the process of converting each word in the corpus to it's corresponding root stem of the English Language. This is done in order to feed the model about the base information that particular word is trying to convey in the context, thus allowing for simplicity for any ML model to generalise better by capturing the interesting patterns and features in the corpus. It has a great impact on retrieval model:

1. It reduces the likelihood of missing relevant documents due to differences in word forms.
2. Stemming enables better matching between query terms and document terms by normalizing morphological variations. For example, a search for "play" could match documents containing "players" or "playing," improving the chances of finding relevant content.
3. Stemming reduces the number of distinct words (tokens) that need to be indexed. This leads to smaller index sizes, which can enhance the speed and efficiency of search algorithms, reducing storage costs and retrieval time.
4. In some cases, stemming can cause ambiguity. Words that share the same root may have different meanings (e.g., "operate" and "operation"). Over-stemming can hurt the relevance of search results by introducing noise, especially in complex languages or domains with specialized vocabularies. However, in general it is observed that it significantly improves the performance and it's advantages outweight it's disadvantages.

Stopwords removal:

1. Stopwords tend to occur frequently across documents but contribute little to distinguishing the relevance of documents. By removing them, the index size of the IR system is reduced, leading to more efficient storage and faster retrieval times.
2. Queries can be processed more quickly since fewer words need to be matched, improving the system's overall speed and responsiveness.
3. Stopwords generally carry low semantic content and do not help in determining the relevance of documents. By removing them, the system focuses on more informative terms. Removing stopwords can improve the effectiveness of TF-IDF (Term Frequency-Inverse Document Frequency) weighting by preventing common but uninformative words from having an undue influence on the scoring of documents or a similar model like Boolean Retrieval Model where we are only forming a subspace depending upon the p-most occurring stems in the corpus, where the information is heavily dependent on frequencies of useful though rarer words.

Insights gained from the document clustering step:

1. TF-IDF: In case of TF-IDF Matrix, we see that there are no well-defined cluster boundaries, and there are many outliers which hamper the clustering quality. The clustering quality metric (Silhouette Score) = 0.049669486334172244 in this case, which means that the clustering is very poor.
2. BM25: In case of BM25 Matrix, we see that there are well-defined cluster boundaries, and there are not many outliers which hamper the clustering quality. The clustering quality metric (Silhouette Score) = 0.15795222874977038 in this case, which means that the clustering is better than that performed by TF-IDF metric. This clearly shows that in case of BM25, we have that the clusters formed are better than TF-IDF. Clustering quality is still not that good because we have not yet accounted for the order of the words in the corpus. Because of this Bag of Words assumption, the Clustering quality is a bit disturbed.

So, we see that both TF-IDF and BM25 try to cluster documents based upon the information conveyed by the respective matrices but fail miserably to do so. The solutions to these problems were developed over a period of time, by the use of

Word2Vec and Transformers, which also look at the neighbouring words, which lead to improved performance.