Implementation of BM25 Algorithm for Search Engine Development of app.rekmed.com

Haikal Muhammad Zahid Ghiffari
(21/472762/PA/20332)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
haikal.muhammad.zahid.ghiffari@mail.ugm.ac.id

Muhammad Rifki Aprinanda Putra
(21/477836/PA/20707)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
muhammad.rifki.aprinanda.putra@mail.ugm.ac.id

Veronika Regina Shanty Octaviani P.
(20/454543/PA/19574)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
veronikaregina@mail.ugm.ac.id

William Hilmy Susatyo
(21/472585/PA/20308)
Department of Computer Science and Electronics, Universitas Gadjah Mada
Building C, 4th Floor North
Yogyakarta, Indonesia
william.hilmy.susatyo@mail.ugm.ac.id

***Abstract*— The important task in information retrieval is to create a ranked retrieval system for a particular objective. On the other hand, a ranked retrieval system is also crucial in the context of a medical platform due to its efficiency, compliance, and ability to prioritize critical information. In this research, we utilize the BM25 (Best Matching 25) algorithm as a basis for the ranked retrieval system that was created from the database of app.rekmed.com, an integrated medical platform. The developed ranked retrieval focuses on three datasets comprising information about doctors, patients, and treatments. Evaluation using various performance metrics indicates that the results are extremely dependent on the input query provided by the end user.**

***Keywords*— Information Retrieval, Database, Document Length Normalization, Text Retrieval Systems, Medical Platform**

I. Introduction

Medical platform is essential in the contemporary healthcare landscape due to its capacity to streamline communication, enhance patient care, and facilitate efficient data management. According to a report by Statista, the global digital health market is expected to reach $170.2 billion by the end of 2023, underscoring the growing significance of technological interventions in healthcare [1]. With features such as electronic health records (EHRs) and telemedicine services, medical platforms offer accessibility to medical information and remote consultations, promoting patient engagement and adherence to treatment plans.

Information retrieval, in general, refers to the process of obtaining information relevant to a user's needs from a large collection of data [2]. The main task of the information retrieval in building a ranked retrieval system is to prioritize and present the retrieved information in a manner that reflects its relevance to the user's query. This involves employing algorithms that assess and rank documents based on factors such as keyword relevance, document quality, and user preferences, ultimately ensuring that the most pertinent results appear at the top of search outcomes.

In the context of medical platforms, where extensive databases often lead to information overload, ranked retrieval systems are considerably instrumental. By prioritizing relevant data and streamlining access, this system addresses the complexities of medical queries, facilitating more informed decision-making by healthcare professionals. The implementation of a ranked

retrieval system not only enhances efficiency in accessing critical medical data but also contributes to improved decision outcomes and compliance in the healthcare sector.

In this study, the BM25 algorithm serves as the foundation for a ranked retrieval system within the app.rekmed.com medical platform database. This retrieval system is designed to address various tasks related to doctors, patients, and treatments, including searching for relevant information within these categories. Users can explore connections such as finding a doctor for a specific patient, identifying patients who underwent a particular treatment, discovering treatments offered by a specific doctor, and determining doctors who have undergone specific treatments.

II. Dataset

The dataset that is being used in this study was obtained from app.rekmed.com, which could be accessed on this link. In this research, we only focus on three datasets, namely the dataset of the patient, doctor, and treatment which indicated respectively in Table 1, Table 2, and Table 3. The column in these dataset are organized as follows:

Table 1. *Dataset of patient*

No	Column	Data Type
1.	*mr*	integer
2.	*klinik_id*	integer
3.	*tanggal lahir*	object
4.	*jk*	object
5.	*alamat*	object
6.	*no_telp*	integer
7.	*pekerjaan*	object
8.	*penanggung_jawab*	object
9.	*foto*	object
10.	*created*	object
11.	*modified*	object
12.	*user_input*	object
13.	*user_modified*	object

Table 2. *Dataset of doctor*

No	Column	Data Type
1.	*user_id*	integer
2.	*nama*	object
3.	*no_telp*	float
4.	*no_telp2*	float
5.	*spesialis*	float
6.	*waktu_praktek*	object
7.	*foto*	object
8.	*alamat*	object
9.	*tanggal_lahir*	object
10.	*created*	object
11.	*email*	object
12.	*alumni*	object
13.	*pekerjaan*	object
14.	*provinsi_id*	float
15.	*kota_id*	float
16.	*jenis_kelamin*	object

Table 3*. Dataset of treatment*

No	Column	Data Type
1.	*id*	integer
2.	*rm_id*	integer

3.	*tindakan_id*	integer
4.	*nama_tindakan*	object

Apart from this, there is also the dataset of medical records that consists of 23 columns, with features such as blood pressure, respiration rate, temperature, weight, and height. Each of the dataset comprises 100 rows, with 12 unique treatments in the dataset of treatment, 43 unique patient names, and 9 unique doctor names.

III. Methods

A. Data Acquisition

Since it is considered as part of the website database, all of the dataset was stored in *.sql* format. Hence, in order to streamline the process, the first step that needs to be done is convert the corresponding dataset, which consists of the dataset of the doctor, patient, treatment, and medical record dataset into tabular form in *.csv* format.

B. Pre-Processing

In the pre-processing step, there are several parts of the dataset that are merged with other dataset in order to achieve the given task. In this case, we started with the dataset of medical records since it has a sharing column with the dataset of patient, doctor, and the treatment. Initially, the dataset of the medical record was merged with the dataset of patient based on the *mr* column, and subsequently merged again with the dataset of the doctor based on the value of *user_id* column to add the information related with the doctor. After that the resulting dataset was merged again with the treatment dataset according to the value of *rm_id* to enhance the dataset with the information of the treatment. For all of the merging process, the left join operation is used since we want to preserve all of the records in medical records dataset whether the patient, doctor, or treatment dataset might contain missing values on certain columns

The next step that is performed in this research is to remove all columns except the name of the patient, doctor, and treatment in the dataset since we only need that information to create the ranked retrieval. At this point, there are missing values that exist in all of the columns. In order to handle these problems, we utilize the forward imputation method to remove the corresponding missing value. This method works by filling that missing value with the value of the last observation [3]. Lastly, all of the words contained in the dataset are converted into lowercase letters to standardize the text data and avoid duplicate representations of words with different cases [4].

C. Exploratory Data Analysis

The visualization that is performed in this step includes the distribution of the missing value in the initial dataset, distribution of the patients, doctor, and treatment, and also the visualization on the resulting sparse matrix based from the implementation of the TF-IDF Method to the dataset. As indicated in Figure 1, the number of missing values in the *patient* column is greater than in the *doctor* and *treatment,* which also directly proportional to the number of unique values, where the number of unique values in *patient* column is higher than other columns.

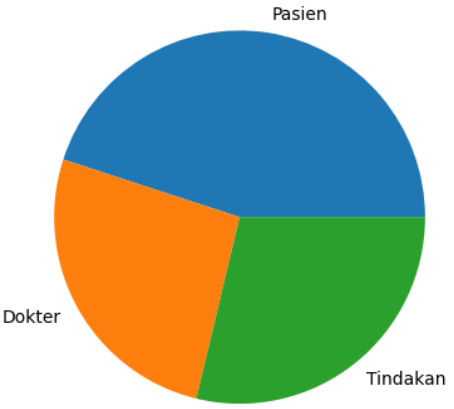

Figure 1. *Distribution of the missing value in the Initial Dataset*

Based on the visualization that depicted in Figure 2, it could be seen that there are two names that often appear in the medical records, consisting of “Dodi Prasetyo” and “Sri Wahyuni”.

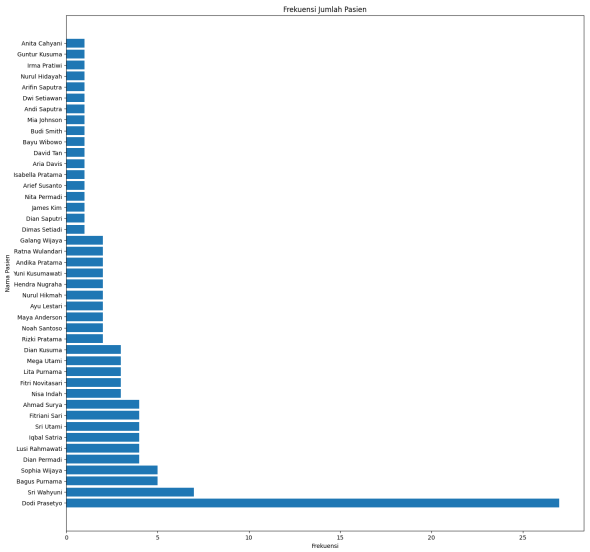

Figure 2. *Distribution of the patients*

According to the Figure 3, "Dr Yudhistya SpOG", "Dr Yoshua", and "Yudhistya" appears many times compared with other names. However, there are several names that seem to be duplicated such as "dr Yudhistya SpOG" and "Yudhistya", which refer to the same person. Apart from that, there is also a doctor with the name "abs" that considered as not meaningful.

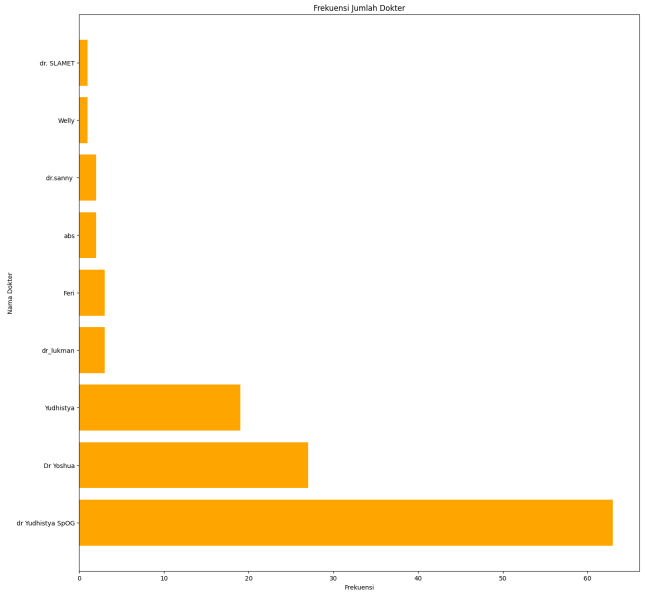

Figure 3. *Distribution of the doctor*

As illustrated in Figure 4, the "Umum" is considered as the treatment that recorded most of the times in the dataset, followed by "USG" and "Konsultasi". In addition, there are also several treatments in the records that have minuscule appearance, namely "asam urat", "kolesterol", and "injeksi". In the further section, these sections will be merged into one category considering its diminutive presence which might imply the imbalance situation.

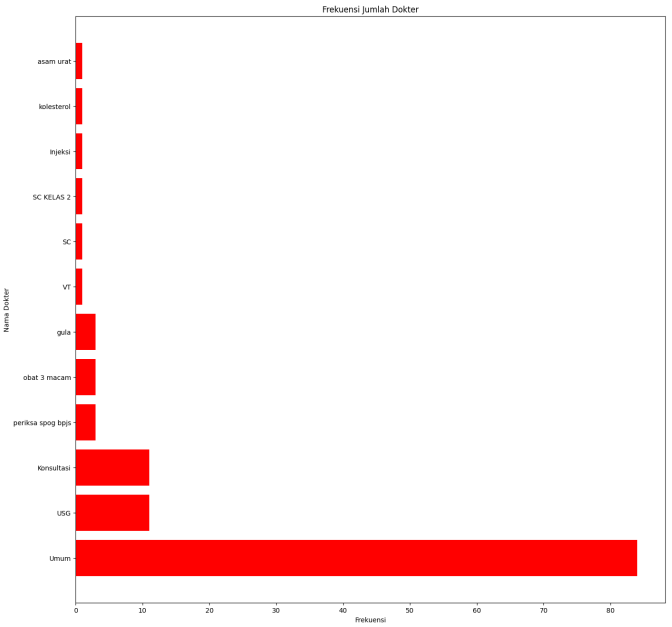

Figure 4. *Distribution of the treatment*

In the Figure 5 depicted below, the TF-IDF method, which works by assessing the importance of a word in a document relative to a collection of documents (corpus) [5], is implemented to the newly created column that concatenates the value in three existing columns, and give the sparse matrix as the output with column represent all of the word in the corpus, and row represent the order of the documents. The value of each of its elements could be 0, which indicates that the particular word doesn't exist in a certain document, or 1, which indicates the opposite. This sparse matrix is then visualized into square representation with blue and white area. The blue and white area represent the element with value 1 and 0 respectively. It could be observed that the total area that is colored blue is almost equal with the area that is colored white, which gives the conclusion that TF-IDF-based method is a suitable method to create the numerical representation of the text since there is no imbalance occurring on the sparse matrix.

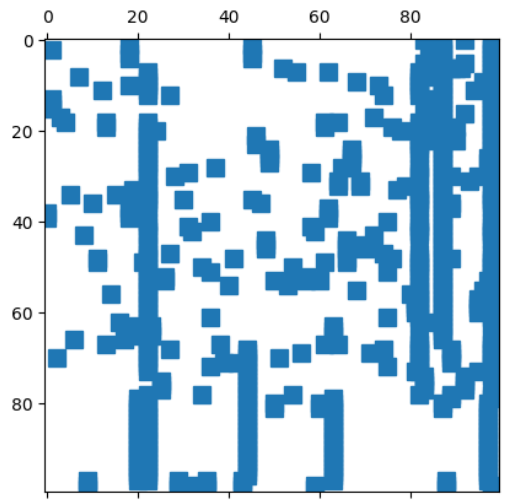

Figure 5. *Visualization of Sparse Matrix of TF-IDF*

D. Feature Engineering

Based on the result of the Exploratory Data Analysis (EDA), especially on the visualization of the treatment distribution, it turns out that there are values that have minuscule appearance while there are also values that appear very often. For those values with minuscule appearance, we merge them into one category, resulting in less cardinality. Beside that, for the duplicated value, we also merge them into one category. In addition, the irrelevant column that appeared in the dataset was treated the same as missing values, where the values were replaced with the values in the previous observation. Subsequently, a new feature represented by the *gabungan* column was also created in this step, which was based on the concatenation of three existing columns, namely the *patient, doctor,* and *treatment* column. This column will be utilized as the basis for the implementation of the model which will be further explained in the following section.

E. Algorithm Implementation

In this research, the BM-25 (Best Matching 25) is utilized as the model to create the ranked retrieval system. BM25 is a ranking function used in information retrieval to score the relevance of documents to a given search query. It's an extension of the TF-IDF (Term Frequency-Inverse Document Frequency) model and introduces additional parameters to better capture the nuances of document relevance [6]. The advantage of BM25 lies in its ability to handle varying document lengths, term saturation, and to provide a more sophisticated ranking compared to traditional TF-IDF, making it particularly effective for search engine applications.

The formula of the TF-IDF could be seen on Equation (1)

$$score\,(q,d) \;=\; \sum_{i=1}^{|q|} idf(q_i) \cdot \frac{tf(q_i,d)\cdot(k_1+1)}{tf(q_i,d)+k_1\cdot\left(1-b+b\cdot\frac{|d|}{avgdl}\right)} \qquad (1)$$

with q represent the inputted query and d represent the particular documents, while k and b respectively depict the saturation parameter (regulates the impact of term frequency), and impact of document length normalization. In specific, it tokenize the inputted query and *gabungan* column in the dataset and then compute the BM-25 Score, before sort it from the highest to the lowest and display the *patient*, *doctor,* and *treatment* column that corresponds with *gabungan* column as could be seen on Figure 6.

```
Input the Query: yudhistya
Input the number of top docs that you want to retrieve: 10
```

	Patient	Doctor	Treatment	Score
0	sri wahyuni	yudhistya	umum	0.997219
1	ahmad surya	yudhistya	umum	0.997219
2	lita purnama	yudhistya	umum	0.997219
3	sri wahyuni	yudhistya	konsultasi	0.997219
4	andi saputra	yudhistya	umum	0.997219
5	sri wahyuni	yudhistya	umum	0.997219
6	ahmad surya	yudhistya	umum	0.997219
7	ahmad surya	yudhistya	umum	0.997219
8	dian kusuma	yudhistya	umum	0.997219
9	sri wahyuni	yudhistya	umum	0.997219

Figure 6. *Sample Implementation of TF-IDF*

IV. Results and Evaluation

A. Results

In the system that has been developed in this study, there are five functionalities that being provided, namely:

1. **Search anything related to the doctor, patient, and treatment**

   This functionality works like a search bar where it might receive any input. The output will be anything that is related to *doctor*, *patient*, and *treatment* columns in the dataset. It will sort the output based on the highest BM-25 Score, as indicated in Figure 7.

```
The ranked retrieval system that have been developed consist of as follows:
1. Search anything related to the doctor, patient, and treatment
2. Search the doctor that handles particular patient
3. Search list of patients that took particular treatment
4. Search list of treatment that could be done by particular doctor
5. Search list of doctor that took particular treatment
What do you want?1
Input the Query: pasien bernama dodi
Input the number of top docs that you want to retrieve: 10
         Patient     Doctor   Treatment     Score
0  dodi prasetyo  dr_lukman        umum  1.453324
1  dodi prasetyo  dr_lukman        umum  1.453324
2  dodi prasetyo  dr_lukman        umum  1.453324
3  dodi prasetyo  dr yoshua  konsultasi  1.333225
4  dodi prasetyo  dr yoshua  konsultasi  1.333225
5  dodi prasetyo  dr yoshua  konsultasi  1.333225
6  dodi prasetyo  dr yoshua  konsultasi  1.333225
7  dodi prasetyo  dr yoshua  konsultasi  1.333225
8  dodi prasetyo  dr yoshua  konsultasi  1.333225
9  dodi prasetyo  dr yoshua  konsultasi  1.333225
```

Figure 7. *Sample Implementation of (1)*

2. **Search the doctor that handles particular patient**

   In this functionality, we need to provide the exact name of the patient that we want to search. The ranked retrieval system will then obtain the list of the name of the doctor from the docs with the highest score with tokenized input of the patient accordingly, as could be seen in Figure 8.

```
The ranked retrieval system that have been developed consist of as follows:
1. Search anything related to the doctor, patient, and treatment
2. Search the doctor that handles particular patient
3. Search list of patients that took particular treatment
4. Search list of treatment that could be done by particular doctor
5. Search list of doctor that took particular treatment
What do you want?2
Input the name of patient: pasien bernama sri
Input the number of top docs that you want to retrieve: 10
             Doctor     Score
0         yudhistya  2.634356
1         yudhistya  2.634356
2         yudhistya  2.634356
3              feri  2.634356
4              feri  2.634356
5              feri  2.634356
6         yudhistya  2.634356
7  dr yudhistya spog  2.232198
8  dr yudhistya spog  1.710080
9        dr. slamet  0.000000
```

Figure 8. *Sample Implementation of (2)*

3. **Search list of patients that took particular treatment**

   In this case, by giving the input of the treatment name, the system will first check whether that treatment exists in the current dataset or not. If that is the case, it will tokenize the input and produce the name of the patient from the docs that has the highest BM25 score with the inputted treatment, as illustrated in Figure 9. Otherwise, it will return the message that the inputted patient doesn't match with all of the records in the patient dataset.

```
The ranked retrieval system that have been developed consist of as follows:
1. Search anything related to the doctor, patient, and treatment
2. Search the doctor that handles particular patient
3. Search list of patients that took particular treatment
4. Search list of treatment that could be done by particular doctor
5. Search list of doctor that took particular treatment
What do you want?3
Input the name of treatment: pelayanan umum
Input the number of top docs that you want to retrieve: 5
     Patient     Score
0  yudhistya  0.997219
1  yudhistya  0.997219
2        abs  0.997219
3  yudhistya  0.997219
4  dr_lukman  0.997219
```

Figure 9. *Sample Implementation of (3)*

4. **Search list of treatment that could be done by particular doctor**

   Similar with the previous functionality, the system received inputted doctor before tokenized, and utilized BM25 Algorithm to gather the treatment from the docs that has the highest score with the tokenized doctor input. The process of the aforementioned functionalities is depicted in Figure 10.

```
The ranked retrieval system that have been developed consist of as follows:
1. Search anything related to the doctor, patient, and treatment
2. Search the doctor that handles particular patient
3. Search list of patients that took particular treatment
4. Search list of treatment that could be done by particular doctor
5. Search list of doctor that took particular treatment
What do you want?4
Input the name of doctor: dokter lukman
Input the number of top docs that you want to retrieve: 5
                                Treatment     Score
0                   umum periksa dokter  2.844587
1  obat 3 macam konsultasi dokter umum  2.511908
2  obat 3 macam konsultasi dokter umum  2.511908
3                                 umum  0.000000
4                                 umum  0.000000
```

Figure 10. *Sample Implementation of (4)*

5. **Search list of doctor that took particular treatment**

   The treatment that is taken as input for the query has to be contained in one of the unique values of the treatment.

Subsequently, the system will tokenize the query and produce the doctor from the docs that has the greatest BM25 score with the tokenized input query, as expressed in Figure 11.

```
The ranked retrieval system that have been developed consist of as follows:
1. Search anything related to the doctor, patient, and treatment
2. Search the doctor that handles particular patient
3. Search list of patients that took particular treatment
4. Search list of treatment that could be done by particular doctor
5. Search list of doctor that took particular treatment
What do you want?5
Input the name of treatment: konsultasi jantung
Input the number of top docs that you want to retrieve: 5
      Doctor     Score
0       feri  1.029654
1  yudhistya  1.029654
2       feri  1.029654
3       feri  1.029654
4  dr yoshua  0.944566
```

Figure 11. *Sample Implementation of (5)*

B. Evaluation

The performance metrics of the BM25 Algorithm that was used in this research was computed using a combination of Precision@K, Recall@K, F-Score@K, and NDCG@K. Precision@K quantifies the proportion of relevant items among the top K, providing a measure of how well the model identifies truly positive instances, while Recall@K focuses on capturing as many relevant items as possible within the top K, offering insight into the model's ability to retrieve all relevant instances. F-Score@K combines precision and recall at the top K items, giving a balanced measure of the model's performance. In addition, NDCG@K assesses ranking quality by considering the relevance scores of items in the top K positions, with diminishing returns for items further down the list, making it particularly useful in scenarios where the graded relevance of items is crucial.

For simplicity, we set the number of top documents in this case to 10. According to the sample implementation of the aforementioned method as depicted in Figure 12, the precision becomes lower as the top K that is considered becomes greater, while the opposite occurs for the recall. It is because in precision, a larger set of items is considered, and it becomes more challenging for all of them to be truly relevant. Precision is highly sensitive to false positives, and including more items in the evaluation increases the likelihood of including non-relevant ones, leading to a lower precision. On the other hand, recall tends to increase with a larger top K because it measures the ability of the model to capture as many relevant items as possible. Including more items in the evaluation gives the model more opportunities to retrieve additional relevant items, leading to a higher recall.

```
Input the patient name: wahyuni
Input the doctor name: yudhistya
Input the treatment name: usg
The inputted query:  ['wahyuni', 'yudhistya', 'usg']
Most similar docs:  ['sri wahyuni', 'yudhistya', 'umum']
```

	Precision @K	Recall @K	F1 Score @K	NDGC @K
0	The precision is undefined	0.000000	The value of Precision or Recall is 0	0.0
1	1.0	0.333333	0.5	1.0
2	1.0	0.666667	0.8	1.0
3	1.0	1.000000	1.0	1.0
4	0.75	1.000000	0.857143	1.0
5	0.6	1.000000	0.75	1.0
6	0.5	1.000000	0.666667	1.0
7	0.428571	1.000000	0.6	1.0
8	0.375	1.000000	0.545455	1.0
9	0.333333	1.000000	0.5	1.0

Figure 12. *Example of Performance Metrics Evaluation*

V. Conclusions

This research utilizes the BM25 Algorithm to create a ranked retrieval system from the dataset of app.rekmed.com, an integrated medical platform. The proper preprocessing method, such as forward imputation, and feature engineering technique, which includes cardinality reduction, are performed to ensure that the resulting ranked retrieval produces reliable results.

Recent findings suggest that the performance metrics of the method is highly dependent on the given query. Nevertheless, the result from most of the queries indicates that the precision tends to be lower as the document that is taken into consideration becomes greater, while the opposite occurs for the recall. The reason lies in the nature of precision, where a broader set of items is taken into account, making it increasingly difficult for all of them to be genuinely relevant. In contrast, recall tends to rise with a larger top K since it gauges the model's effectiveness in capturing as many pertinent items as possible.

For the future improvements, it is advisable to compare it with other ranked retrieval methods, such as Vector Space Model (VSM) and LambdaMART since it is essential for understanding its strengths and weaknesses, selecting the most appropriate model for specific tasks, and staying informed about advancements in the field of information retrieval.

VI. Appendix

All of the documentation, asset, and the source code of the research that has been done could be seen on the following link

References

[1] Statista, "Digital Health - Worldwide," 2023. https://www.statista.com/outlook/hmo/digital-health/worldwide (accessed November 24, 2023)

[2] Manning, C.D., Raghavan, P., Schütze, H., "An Introduction to Information Retrieval," Cambridge University Press, 2009. https://ds.amu.edu.et/xmlui/bitstream/handle/123456789/14697/Book%20558%20pages.pdf?sequence=1&isAllowed=y (accessed November 25, 2023)

[3] Platias, C., Petasis, G., "A Comparison of Machine Learning Methods for Data Imputation," SETN 2020: 11th Hellenic Conference on Artificial Intelligence (pp. 150-159), 2020. doi.org/10.1145/3411408.3411465

[4] Jakhotiya, A., Jain, H., Jain, B., Chaniyara, C., "Text Pre-Processing Techniques in Natural Language Processing: A Review, " International Research Journal of Engineering and Technology (IRJET) Vol.9, Issue 2 ,2022. https://shorturl.at/fjUW2

[5] Abolghasemi, A., Askari, A., Verberne, S., "On the Interpolation of Contextualized Text-Based Ranking with BM25 for Query-by-Example Retrieval," ICTIR '22: Proceedings of the 2022 ACM SIGIR International Conference on Theory of Information Retrieval (pp. 161-170), 2022. doi.org/10.1145/3539813.3545133