PRI Report - Delivery 1

Group 04
96656 - Joaquim Bação
99258 - Jorge Santos
110946 - Francisco Abreu

1 Introduction

To address the given tasks related to term and document frequencies, it was necessary to select a suitable model for evaluation. Various combinations were tested, considering different preprocessing options as well as multiple Information Retrieval (IR) models, including TF-IDF, BM25, BM25f, and Language Models (LM) with their respective variants. Additionally, the effectiveness of Reciprocal Rank Fusion (RRF) will be discussed in this report.

To guide the iteration improvements, a baseline (TF-IDF model) was established and an evaluation system was put in place utilizing the *pytrec_eval* [3] framework. This utilized metrics such as MAP and NDCG@10 to evaluate the models' document ranking order according to each provided query.

2 Design Choices

As mentioned before in this work we will make decisions that will impact the end results of this work, and this section will describe them.

2.1 Preprocessing

To determine which preprocessing method would provide the best results when evaluating the indexes, four different preprocessing options were tested.

Raw: The "Raw" preprocessing option uses the `RegexTokenizer`, which only tokenizes words without any further modification. This approach serves as the base case, just ensuring that the basic elements of the index are tokens, rather than longer text segments.

Standard: The "Standard" preprocessing uses the Whoosh analyzer to convert tokens to lowercase and remove common stop words (e.g., "the," "is," "and"). This results in a filtered set of tokens, improving search relevance by excluding unimportant words.

Stemming: The "Stemming" preprocessing builds on the "Standard" approach by reducing words to their root form (e.g., "running" becomes "run") using the *Porter* stemmer. This consolidation helps improve search accuracy by recognizing different word forms as the same concept.

Language: The "Language" preprocessing extends "Stemming" by applying language-specific techniques, such as removing or normalizing certain word forms and making use of the *Snowball* stemmer. This ensures better indexing accuracy by addressing language nuances, especially in multilingual contexts.

Each method was tested to analyze its effect on tokenization and indexing, however after concluding the experiments, we choose to proceed in the next steps making use of *Stemming* preprocessing.

2.2 Indexing

The first programming task involved developing a function to preprocess the collection and, using existing libraries, build an inverted index. This data structure is used in information retrieval to efficiently locate documents containing specific keywords or phrases, along with the relevant statistics for the subsequent functions.

Input: The collection of documents to be indexed and the preprocessing strategy to apply, are passed as arguments to the function.

Initialization: As soon as the function starts, a start time is registered and the directory for storing the index files is prepared.

Text Processing and Schema: The schema of the index consists of four fields: **id**, **title**, **abstract** and **date**, which correspond to the document fields being indexed. The text processing step determines which analyzer is applied to the fields of the schema.

Index Type and Creation:

A Whoosh index is created to enable fast and efficient searching and retrieval of documents based on tokenized and processed text. The documents in the dataset are processed by iterating through each document, extracting its title and abstract, and adding them to the index. This structure allows for efficient indexing and retrieval of relevant documents.

Index Time Calculation: The time taken to build the index is calculated by subtracting the start time from the final time.

Index Size Calculation: The index size is calculated by summing the size of all files in the index directory, then converting it to megabytes (MB).

Output: The output of the function includes the created Whoosh index object, the time taken to index the documents, and the size of the index in MB. This information is useful for evaluating the indexing performance and resource usage.

2.3 IR Models

For Information Retrieval models, 6 algorithms were utilized: 4 based on the whoosh scoring framework (TF-IDF, BM25, BM25f, LM), 1 that does not provide ranking of results (Boolean Query) and a results combining ranking algorithm (RRF). We decided to focus the report on the ranking schemes since these provide more palpable results to be evaluated.

Boolean Query: Boolean retrieval is one of the foundational models in Information Retrieval (IR). This method does not rank documents but rather returns a set of documents that strictly match the Boolean expression formed by the query terms. The Boolean model assumes that each term is either present or absent in a document, leading to the retrieval of all matching documents without ranking them by relevance.

TF-IDF: The baseline of this work, the TF-IDF model introduces a weighting scheme that maximizes simple term frequency ranking by reducing the impact of commonly occurring words and giving more weight to rare but significant terms. It is calculated multiplying the Term Frequency (TF) with the Inverse Document Frequency (IDF).

BM25: The BM25 incorporates a probabilistic ranking function and length normalization. BM25 better handles term frequency saturation and gives more balanced weighting to longer documents, leading to improved retrieval performance across various datasets.

BM25F: Utilizing the same type of calculations as BM25, the BM25f differs from the previous by introducing weights to each section of the document, thus allowing for different, more granular analysis.

LMs for IR: Whoosh's PL2 scoring function is a probabilistic language model-based retrieval function derived from Divergence from Randomness (DFR), capable of estimating term importance using Poisson modeling and document length normalization.

Reciprocal Rank Fusion (RRF): RRF combines rankings from multiple retrieval models to mitigate individual biases. It assigns higher weight to top-ranked results while integrating lower-ranked ones, leading to more diverse and fair search rankings.

2.4 Ranking

The ranking process was applied to retrieve and order documents based on their relevance to each query. Different retrieval models, including TF-IDF, BM25, BM25F and LM, were used to generate ranked lists of documents, ensuring a diverse evaluation of ranking strategies. The ranking results were then processed to extract the top 1000 documents per query, forming the basis for subsequent evaluation.

As an extra, we decided to apply a time-based boost [2] for the year of the publication to improve the results of our IR models since most results are only relevant after 2019 (year in which the pandemic started).

To analyze the performance of different ranking methods, we recorded the execution time and memory usage of the ranking process. This allowed us to assess not only the effectiveness of each method but also its computational efficiency.

2.5 Evaluation

The IR system's performance was assessed using `pytrec_eval` [3] with multiple evaluation metrics. A total of 50 queries with 1000 ranked documents per query were used, resulting in the evaluation of 50 000 documents, ultimately, ensuring a comprehensive performance analysis across diverse queries.

The evaluation of the performance of our IR model systems were compared using the following metrics: MAP@1000, NDCG@1000, NDCG@10, and P@10, which assess the relevance and ranking quality of the retrieved documents. These metrics were chosen to provide a comprehensive analysis of the system's effectiveness in returning relevant results, particularly focusing on ranking quality, precision, and recall.

Map@1000: The Mean Average Precision (or Map) captures the average of the precision value obtained for the top k (in this case 1000) documents, each time a relevant document is retrieved.

NDCG@1000 and NDCG@10: NDCG measures ranking quality by discounting relevant documents that appear lower in the ranking. NDCG@1000 evaluates the top 1000 documents, while NDCG@10 focuses on the top 10. Higher NDCG values indicate better ranking, particularly for high-relevance documents.

P@10: Precision at 10 measures the proportion of relevant documents among the top 10 retrieved. It offers a quick view of ranking effectiveness, with higher values indicating better performance in the top ranks.

3 Questions to Explore

3.1 Characterize the Document Collection D and Topic Collection Q

The corpus D consists of 192,509 documents. The Fig. 1 shows that the majority of documents were published between 1980 and 2021, with a peak during the pandemic.

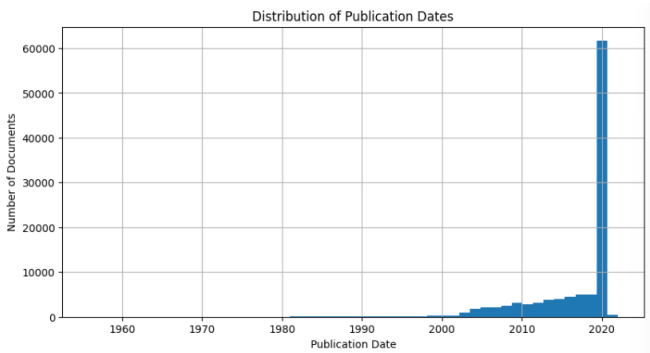

Figure 1: Document Publication Date Distribution.

The dataset captures a **temporal evolution of research**, which is useful for *time-based retrieval* (as we will see), providing insights into trends and emerging research areas over time.

3.1.1 Characterizing the Document Collection (D)

	title_length	abstract_length
count	192509	192509
mean	12.609	133.038
std	5.935	133.038
min	0	0
max	146	18000

Table 1: Summary statistics of title and abstract lengths.

Titles have an average length of 12.6 words, with a maximum of 146 words. Abstracts, on the other hand, show a larger variance, with some being very concise while others contain extensive text up to 18,000 words. An histogram, Fig. 13 in the appendix, further shows that most abstracts are within 0 to 500 words, with some outliers containing detailed and extensive information.

3.1.2 Characterizing the Topic Collection (Q)

The topic collection contains 50 queries, structured into (title, description, and narrative). **Query Length Distribution:** The title field (i.e., the keyword query) varies in length, with most containing 2-5 words (Fig 2). Fig 3 shows the most frequent terms in queries, with "coronavirus" and "COVID-19" dominating the distribution.

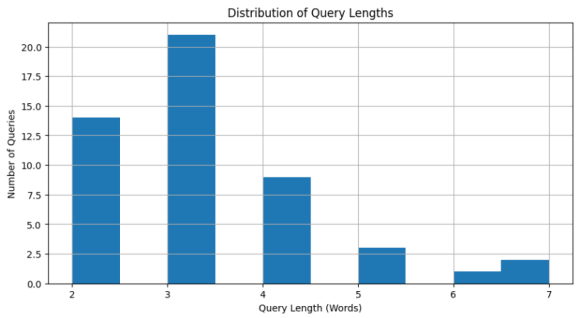

Figure 2: Distribution of Query Lengths.

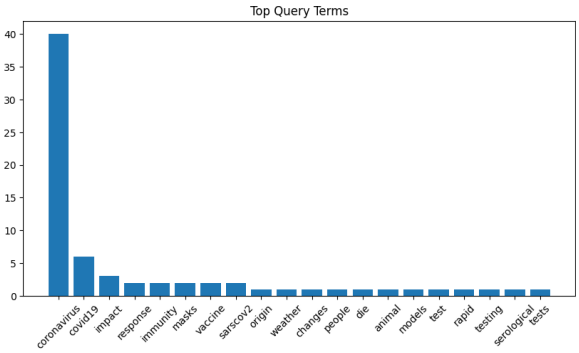

Figure 3: Top Query Terms.

3.1.3 Text Processing Steps

The text processing workflow consisted of multiple steps, aimed at standardizing and cleaning the dataset for more effective analysis:

- **Tokenization**: This step breaks sentences into individual words (tokens), making them easier to analyze.
- **Lowercasing**: Converts all text to lowercase to ensure uniformity and prevent duplicate words (e.g., "COVID" vs. "covid").
- **Stopword Removal**: Removes frequent but non-informative words like "the", "of", and "and", which do not add meaning in an information retrieval context.
- **Punctuation Removal**: Eliminates punctuation marks such as commas, periods, and brackets, reducing noise in word frequency calculations.

Before Processing: The raw term frequency is shown in Figure 4. The most frequent terms include stopwords and punctuation, which contribute to noise in retrieval tasks.

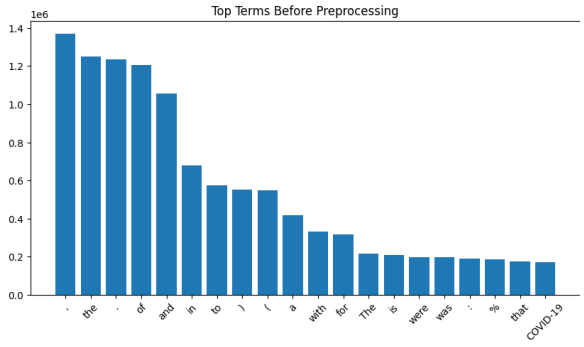

Figure 4: Top Terms Before Processing.

After Processing: After removing stopwords and punctuation, the dataset became more refined and focused on biomedical concepts. The most frequent words now relate directly to the core subject matter.

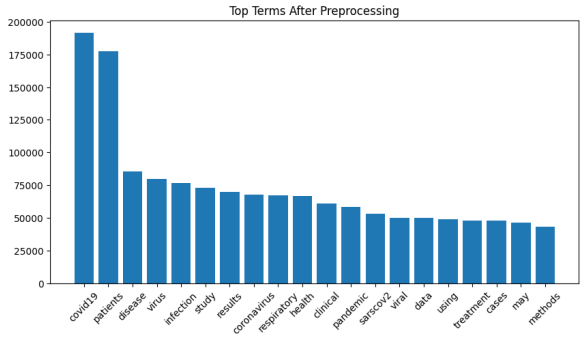

Figure 5: Top Terms After Processing.

Fig. 12 in the appendix, a word cloud, visually highlights the most relevant medical terms appearing in the dataset.

3.1.4 Overlapping Top Terms Among Q Topics

Lexical Overlaps: Queries and documents share key terms, including "biomarkers", "drug", "transmission", "symptoms", and "serological".

Semantic Overlaps: Using word embeddings based on Word2Vec reveals that query terms have meaningful similarities within the dataset:

- **biomarkers**: biomarker, markers, signatures
- **analysis**: analyses, study, comparisons
- **datasets**: dataset, classifiers, algorithms

This suggests that query expansion techniques could significantly enhance retrieval performance by incorporating semantically related terms.

3.1.5 Conclusions

1. The dataset provides a **rich collection of biomedical research**, with titles and abstracts varying in length and detail.
2. **Preprocessing significantly improves term relevance**, eliminating noise and highlighting domain-specific concepts.
3. **Query-document overlap is strong**, meaning simple keyword retrieval is effective, but *semantic expansion could further improve results*.
4. Future work should explore **enhanced ranking models** such as Word2Vec-assisted retrieval.

3.2 Characterize the performance of the IR system

3.2.1 Indexing Performance Across Preprocessing Methods

To evaluate the space and time requirements for processing and indexing, the four preprocessing methods were tested again: raw (no analyzer), standard (removes stop words and converts all tokens to lowercase), stemming (which adds stemming to the standard method using Porter) and language (which adds stemming to the standard method using Snowball). The indexing times and index sizes were recorded and are presented in the graph below (Figure 6).

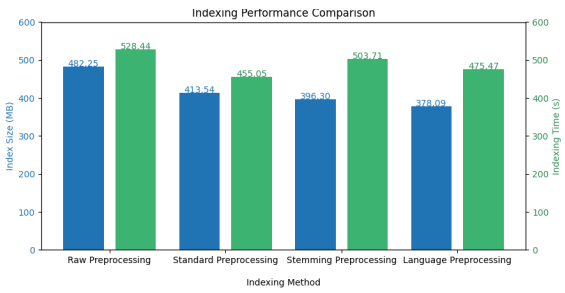

Figure 6: Indexing Performance Across Preprocessing Methods.

Raw indexing requires the most space and time because every word is indexed as a separate token without any filtering, leading to higher storage and processing costs. Standard indexing improves efficiency by applying basic preprocessing, which reduces the index size compared to raw indexing and makes processing faster.

Stemming further reduces space usage by converting words to their root forms, eliminating variations of the same word. However, this extra computational step makes it slower than standard indexing. Language-specific preprocessing achieves the smallest index size by applying

more advanced linguistic rules but requires slightly more processing time than standard indexing, though it remains faster than stemming.

3.2.2 Ranking Performance Across Retrieval Models

To evaluate the space and time required for the retrieval stage, we measured the memory usage and execution time of different ranking models. The memory usage accounts for the extra RAM required to perform ranking, while execution time measures how long each method takes to retrieve results. All methods were tested only on the stemming-preprocessed indexes, as the comparison aimed to evaluate the performance differences between the retrieval models rather than the effect of preprocessing. Fig. 7 presents these metrics for TF-IDF, BM25, BM25F and LM.

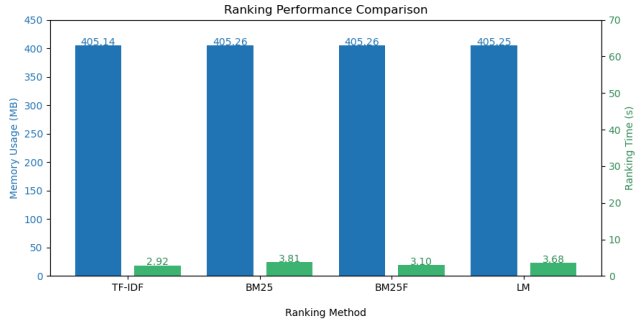

Figure 7: Ranking Performance Across Retrieval Models.

Memory usage remained constant across all models, indicating that the ranking function's RAM consumption was not significantly impacted by the choice of retrieval model. However, execution time showed slight variations. TF-IDF completed the ranking process faster than BM25 models and LM, which took the same amount of time. This difference can be attributed to the additional computations of these models, such as document length normalization and parameterized weighting functions, which introduce more complexity compared to the simpler frequency-based calculations of TF-IDF.

3.3 Given a specific *p*, is the implemented IR system better at providing recall or precision guarantees?

To determine whether the system is better at promoting precision or recall with a given number of p documents, we compared the results of the execution with the referenced ones by *qrels*. Fig. 8 shows that the recall increases with the number of documents being considered in the ranking (the bigger the number of documents considered, the bigger the probability a relevant document is considered, assuming the system is not perfect), whilst the precision presents the opposite trend, with the number of documents considered increasing reflecting a decrease in the system precision (the bigger the number of documents, the higher the probability more irrelevant documents are considered ranked). Overall, these results are in accordance with the expected trends for these metrics.

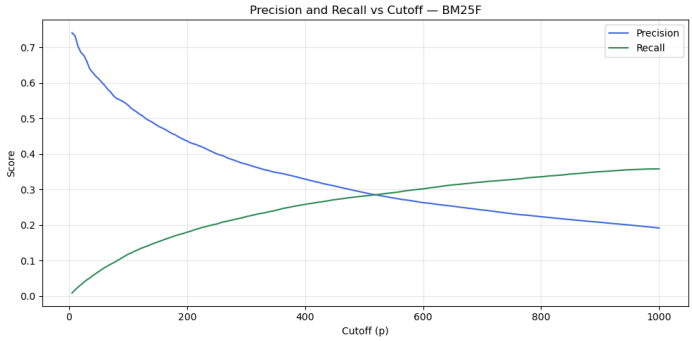

Figure 8: Precision and Recall values as p increases.

3.4 Which p should be used by the IR system if the user has a preference towards: minimizing false positives, minimizing false negatives, or maximizing true positives?

The choice of p (the number of retrieved documents) significantly impacts the trade-off between **precision** and **recall**. Depending on the user's objective, the IR system should adjust p accordingly:

Minimizing False Positives (FP) – Prioritizing Precision: False positives occur when an *irrelevant* document is retrieved. Users who cannot tolerate incorrect documents, such as **clinical researchers requiring high-confidence papers**, should prioritize precision. Using **smaller values of p (e.g., 10, 20, 50)** ensures that only the most relevant documents are retrieved.

The best evaluation metrics for this scenario are **Precision@10 (P@10) and nDCG@10**, which measure the relevance of top-ranked results. Retrieval models such as **BM25F and TF-IDF** are best suited for precision-focused ranking.

Minimizing False Negatives (FN) – Prioritizing Recall: False negatives occur when a *relevant* document is not retrieved. This is critical for **medical researchers requiring comprehensive literature coverage on COVID-19**. To avoid missing relevant documents, the system should use **larger values of p (e.g., 500, 1000)**.

Mean Average Precision (MAP) is a useful metric for balancing recall and ranking effectiveness, while **Recall@1000** measures the proportion of relevant documents retrieved. Retrieval models such as **PL2 (Language Model) and BM25F** work well for recall-oriented retrieval.

Maximizing True Positives (TP) – Balancing Precision and Recall: True positives refer to *retrieved documents that are actually relevant*. To maximize TP, the IR system should balance both precision and recall, using an **intermediate value of p (e.g., 100, 200, 500)**.

nDCG@10 and MAP are the best evaluation metrics for this trade-off. A hybrid approach using **BM25F combined with Reciprocal Rank Fusion (RRF)** can further enhance the ranking by integrating multiple retrieval models.

A summary of these conclusions can be seen in Table 6 in the appendix.

3.5 Performance Variation Across Queries

Evaluating the retrieval effectiveness of our system requires not only assessing overall performance but also understanding how performance varies across different queries. Some queries may consistently yield high-quality results, while others perform poorly due to factors such as ambiguity, topic complexity, or limitations in the ranking function. To investigate this, we analyzed per-query performance using key evaluation metrics.

The goal of this analysis is to determine whether performance significantly varies across queries and, if so, to identify which queries or topics exhibit lower retrieval effectiveness. By computing metrics like MAP@1000, NDCG@1000, NDCG@10, and P@10 on a per-query basis, we can measure the degree of variability in retrieval effectiveness. Additionally, computing summary statistics such as mean, standard deviation, and minimum/maximum values allows us to quantify the consistency of the ranking function.

After running evaluation per query, these were the results:

Model	Mean	Std Dev	Min	Max
TF-IDF	0.5020	0.3228	0.0000	1.0000
BM25	0.7080	0.3104	0.0000	1.0000
BM25F	0.7320	0.2908	0.0000	1.0000
LM	0.7100	0.3048	0.0000	1.0000

Table 2: P@10 - Model Evaluation Metrics

Model	Mean	Std Dev	Min	Max
TF-IDF	0.4413	0.2974	0.0000	0.8611
BM25	0.6631	0.3016	0.0000	1.000
BM25F	0.6687	0.2936	0.0000	1.0000
LM	0.6588	0.3006	0.0000	1.0000

Table 3: nDCG@10 - Model Evaluation Metrics

Model	Mean	Std Dev	Min	Max
TF-IDF	0.2615	0.1437	0.0142	0.5696
BM25	0.3773	0.1845	0.0199	0.7176
BM25F	0.3890	0.1824	0.0180	0.6915
LM	0.3823	0.1972	0.0102	0.7587

Table 4: nDCG@1000 - Model Evaluation Metrics

Model	Mean	Std Dev	Min	Max
TF-IDF	0.0960	0.0885	0.0010	0.3129
BM25	0.1916	0.1427	0.0008	0.5247
BM25F	0.2021	0.1453	0.0006	0.5130
LM	0.1958	0.1513	0.0003	0.5419

Table 5: MAP@1000 - Model Evaluation Metrics

From the results, BM25, BM25F and LM consistently outperform TF-IDF across all metrics, with higher mean values and better worst-case performances, reflected by their higher minimum values. All three, the BM25, BM25F and LM achieve similar high maximum values, indicating their potential for high performance on specific queries, while TF-IDF has consistently lower maximum values, which aligns with its overall weaker performance.

In general, BM25, BM25F and LM are more reliable and consistent, making them preferable choices for most use cases, while TF-IDF can still be useful for certain high-performance queries despite its inconsistency.

3.6 How different text processing and scoring options affect retrieval? Is reciprocal rank fusion useful to place ensemble decisions?

How different text processing and scoring options affect retrieval? As previously mentioned in section 2, several configurations were tested to ultimately determine the best model possible for information retrieval. From these we found that the model that maximized the MAP and nDCG@10 metric was the BM25f with language preprocessing (utilizing English) and 3 times the weight of the abstract over the weight of the title (boost="title":1, "abstract": 3). To allow for comparisons, Fig. 9 shows the evaluation results for different models according to the established metrics.

Is reciprocal rank fusion useful to place ensemble decisions? To determine whether Reciprocal Rank Fusion (RRF) is useful for ensemble decision-making, we need to analyze its impact compared to individual ranking models. The goal of RRF is to combine rankings from multiple retrieval models, ensuring that documents ranked highly by any model are promoted in the final list.

It is important to notice that Reciprocal Rank Fusion (RRF) operates on a different scale from LM and BM25

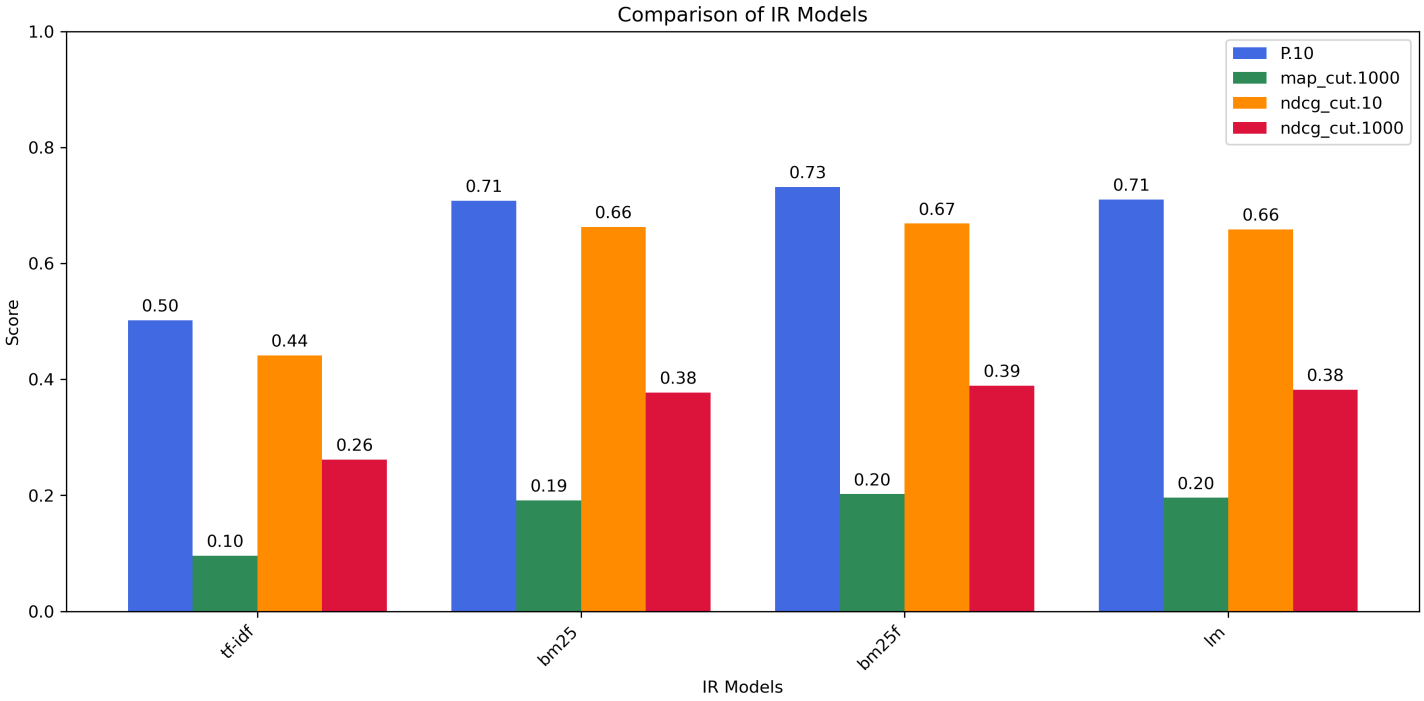

Figure 9: Evaluation of IR Models according to established metrics.

because it is a rank-based fusion method, whereas LM and BM25 are score-based retrieval models.

To further evaluate the effectiveness of RRF, we compare its ranking outputs with those of BM25F and LM (our best models). Fig. 10 presents the document rankings for one of the queries.

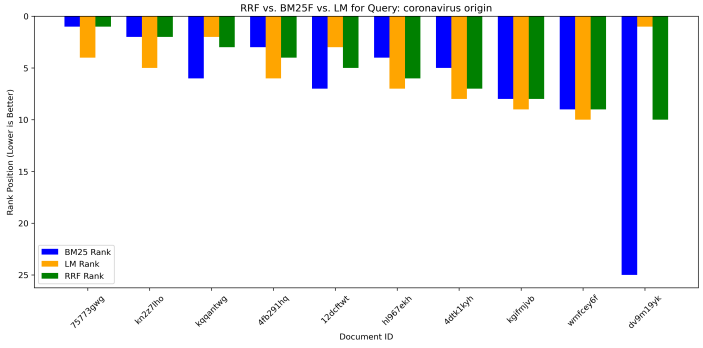

Figure 10: Comparison of ranking between IR models for one query.

From the results, we observe that RRF scores are **closely aligned with BM25F and LM ranks**, indicating that it effectively integrates information from both models. The RRF ranking ensures that documents highly ranked by any model maintain strong positions in the final list.

The equation governing Reciprocal Rank Fusion [1] with k=60 ensures that documents ranked highly by any individual retrieval model receive a boost in the final fused ranking. This is particularly useful when different models highlight different relevant documents. By summing the reciprocal rank contributions across models, RRF helps establish consensus in retrieval, reducing the reliance on a single retrieval method and promoting a balanced ranking.

Furthermore, this approach allows us to assess whether consensus improves retrieval effectiveness or if specific models perform best under certain conditions. If the fused ranking consistently outperforms individual models in retrieval quality metrics such as NDCG and MAP, it is possible to conclude that **fusion is beneficial**, as seen in Fig. 11. Otherwise, if one model consistently dominates, this indicates that a particular retrieval approach is inherently stronger for the given task.

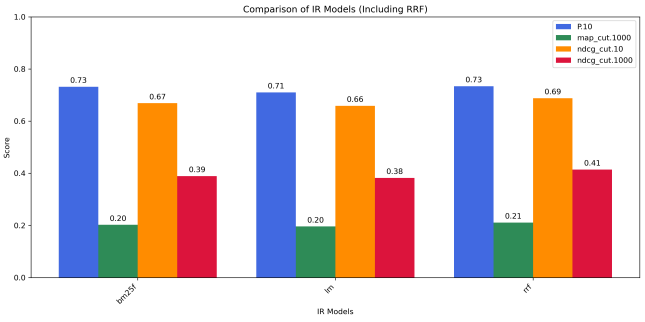

Figure 11: Reciprocal Rank Fusion results compared with BM25F and LM models.

Ultimately, **RRF proves to be an effective technique for ensemble ranking**, balancing contributions from different models while maintaining relevance and diversity in the retrieved results.

4 Conclusion

In this work, we characterized and evaluated a biomedical information retrieval (IR) system using a large document corpus and a set of predefined queries. Our goal was to test and assess the performance of different retrieval models, the impact of preprocessing techniques, and the variation in effectiveness across individual queries.

We found that preprocessing, including tokenization, stopword removal, and stemming, significantly enhanced the quality of the dataset for retrieval. By reducing noise and emphasizing domain-specific terms, we improved the relevance of search results. However, stemming introduced additional computational costs, which, while reducing index size, slowed indexing times compared to simpler preprocessing methods.

For the evaluation of different retrieval models—TF-IDF, BM25, BM25F and LM—the results revealed that BM25, BM25F and LM models generally outperformed others in terms of retrieval quality, providing higher precision and recall. TF-IDF, while faster, was less effective in capturing the nuanced relationships between documents and queries. These results underscore the importance of more sophisticated ranking functions like BM25 and LMs for improving retrieval accuracy.

Overall, the system performed well for general biomedical document retrieval, but fine-tuning retrieval models, preprocessing techniques, and query handling will be crucial for further improvements.

References

[1] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. "Reciprocal rank fusion outperforms condorcet and individual rank learning methods". In: *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '09. Boston, MA, USA: Association for Computing Machinery, 2009, pp. 758–759. ISBN: 9781605584836. DOI: 10.1145/1571941.1572114. URL: https://doi.org/10.1145/1571941.1572114.

[2] Xiaoyan Li and W. Bruce Croft. "Time-based language models". In: *Proceedings of the Twelfth International Conference on Information and Knowledge Management*. CIKM '03. New Orleans, LA, USA: Association for Computing Machinery, 2003, pp. 469–475. ISBN: 1581137230. DOI: 10.1145/956863.956951. URL: https://doi.org/10.1145/956863.956951.

[3] Christophe Van Gysel and Maarten de Rijke. "Pytrec$_e$$val$: $AnExtremelyFastPythonInterfacetotrec_eval$". In: *The 41st International ACM SIGIR Conference on Research Development in Information Retrieval*. SIGIR '18. ACM, June 2018. DOI: 10.1145/3209978.3210065. URL: http://dx.doi.org/10.1145/3209978.3210065.

Appendices

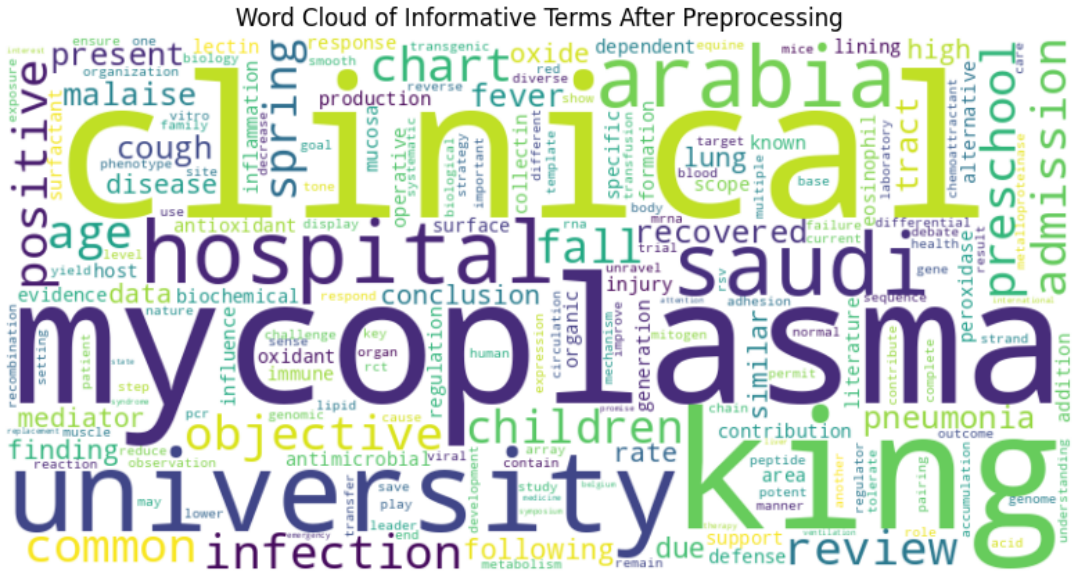

Figure 12: Word Cloud of Informative Terms After Preprocessing.

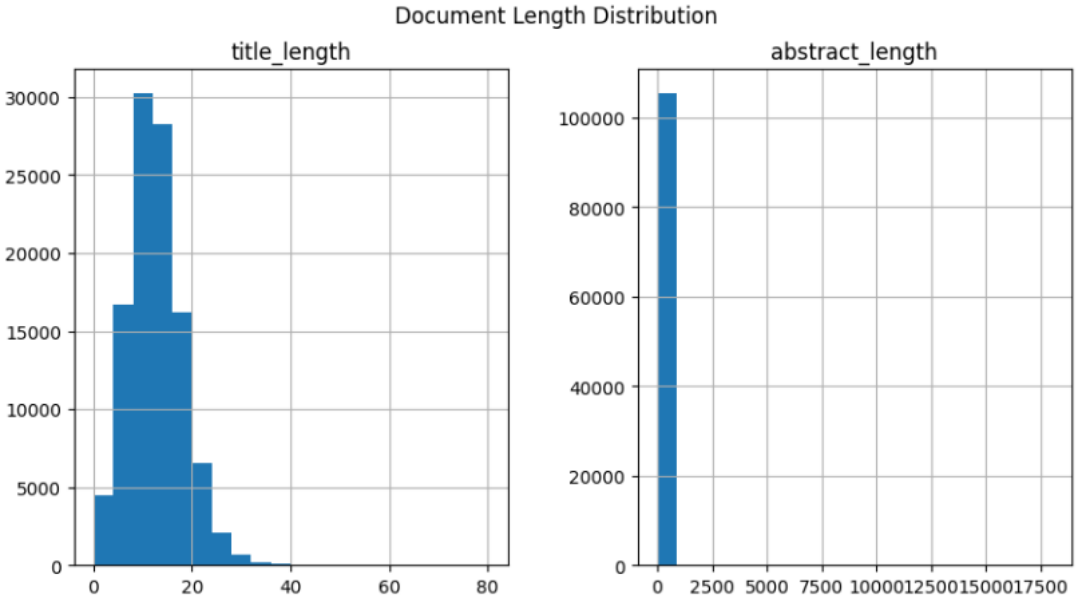

Figure 13: Document Length Distribution.

Goal	Recommended	Best Metrics	Best Models
Minimize False Positives (FP)	p = 10, 20, 50	P@10 nDCG@10	BM25F TF-IDF
Minimize False Negatives (FN)	p = 500, 1000	MAP Recall@1000	PL2 (LM) BM25F
Maximize True Positives (TP)	p = 100, 200, 500	nDCG@10 MAP	BM25F + RRF

Table 6: IR Strategies