SEUPD@CLEF: Team Kalu on improving Search Engine Performance with Query Expansion and Re-Ranking Approach

Kimia Abedini[1], Akan Akysh[1], Arwa Fahoud[1] and Nicola Ferro[1]

[1]University of Padua, Italy

Abstract

This report provides a detailed description of the search engine system designed by Team KALU for the *Conference and Labs of the Evaluation Forum (CLEF)* LongEval LAB 2024 Task 1. The team, composed of students from the University of Padua, developed this system to efficiently index, search, and retrieve documents. We begin by outlining the problem and then go on to describe our system which mainly works on a collection of documents written in French language, then we explain the various methodologies we implemented. We present our experimental results and analyze them according to the techniques we employed. Finally, We present the outcomes of our experiments and discuss the different techniques used.

Keywords

Information Retrieval, Search Engines, Retrieve Documents, Query Expansion, LongEval, CLEF

1. Introduction

Search engines have transformed people's access to information. These systems proved that they stand as fundamental tools in people's daily lives, providing a vast amount of data related to various aspects, from academic research and global news to everyday queries like shopping and local weather. However, the exponential growth in data available online presents a significant challenge for search engines in terms of storage, indexing, and retrieval. This paper introduces a solution to the issue by creating an information retrieval system capable of adjusting to evolving data while sustaining high performance.

Our approach is implementing a search engine to address task 1 of the "LongEval" lab proposed by CLEF 2024, which aims to search a corpus of documents and retrieve the most relevant ones to a predefined set of queries gathered from *Qwant*[1].

The paper is structured as follows: Section 2 shows the related work we have started from; Section 3 outlines our approach; Section 4 explains our experimental setup; Section 5 discusses our main findings; and Section 7 concludes with reflections and future directions.

2. Related Work

We used the paper by LongEval organizers [2] to understand the task and the datasets provided by CLEF. This helped us understand how the documents and queries were collected and what the main objectives of the task were. The paper also provided baseline performances, which we used to benchmark our system's development.

Based on the works of the *CLOSE* [3] and the *FADERIC* [4] teams, we chose to utilize query expansion techniques. We explored various methods, such as different *Large Language Model (LLM)*s and different prompts. Also, We build our re-ranking approach based on the work of JinaAi [5] which developed the jina-reranker model.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 9–12, 2024, Grenoble, France

kimia.abedini@studenti.unipd.it (K. Abedini); akan.akysh@studenti.unipd.it (A. Akysh); arwa.fahoud@studenti.unipd.it (A. Fahoud); nicola.ferro@unipd.it (N. Ferro)

https://www.dei.unipd.it/~ferro/ (N. Ferro)

0000-0001-9219-6239 (N. Ferro)

3. Methodology

In this section, we describe the steps we took to create the different components that comprise the search engine with the different configurations used with each component.

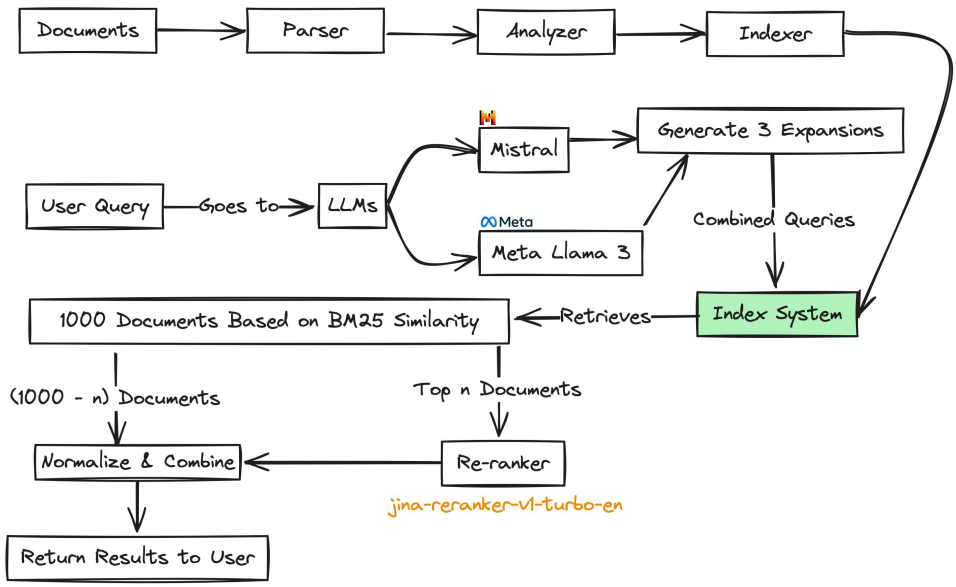


Figure 1: Workflow of the IR system implemented by KALU.

3.1. Parser

Parser processes the collection of documents, extracting valuable information and filtering out irrelevant data. we also use parser to extract the id and body of the documents. our parser consists of three classes which are *ParsedDocument*class, *DocumentParser* class and *LongEvalDocumentParser* class.

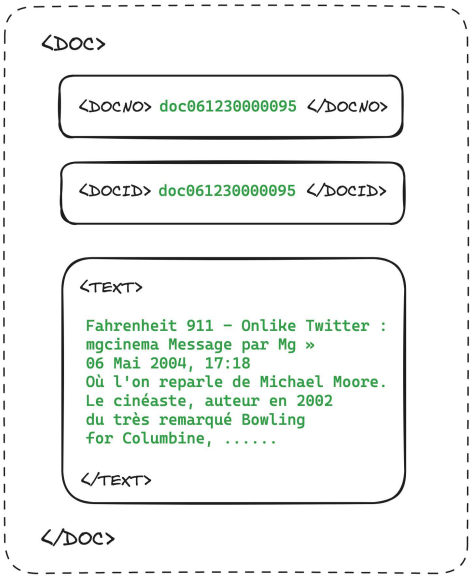


Figure 2: Document structure without parsing.

- **ParsedDocument** class represents a parsed document to be indexed. It has two attributes: ID for the unique identifier of the document and body for the document's content. This class provides functionalities to set and retrieve documents' attributes.
- **DocumentParser** class represents an abstract class providing basic functionalities to iterate over the elements of a ParsedDocument, reading and parsing its content.
- **LongEvalDocumentParser** class is the specific DocumentParser for the LongEval corpus. It provides an implementation of a parser for the documents in the *TREC* format. It reads the document and it replaces all the tags with space and returns *ParsedDocument* that contains the ID and the Body of the document.

3.2. Analyzer

Analyzer is used to apply further processing steps to both the documents and queries. The Analyzer does the functionalities of "Tokenization", "Stemming" and filtering "Stop Words".

- **Tokenization** is the process of splitting text into tokens (each token can be thought as a single word). We have used *StandardTokenizer*[6] in our project, which is one of the tokenizers provided by Apache Lucene.
- **Stemming** is The process of removing suffixes or prefixes from words to obtain their root form. in this system we have tried using *FrenchLightStemmer* [7].
- **Stop Words removal** is the process of eliminating words that frequently appear in most documents and carry little meaningful information for search queries.our stopwords list includes words extracted from *Luke* [8] which is a tool provided by Lucene [9] and also we used the stopword list in *Kaggle* [10]. Removing stopwords improved the search engine's performance by reducing the index size and, consequently, decreasing the time needed to search the index.

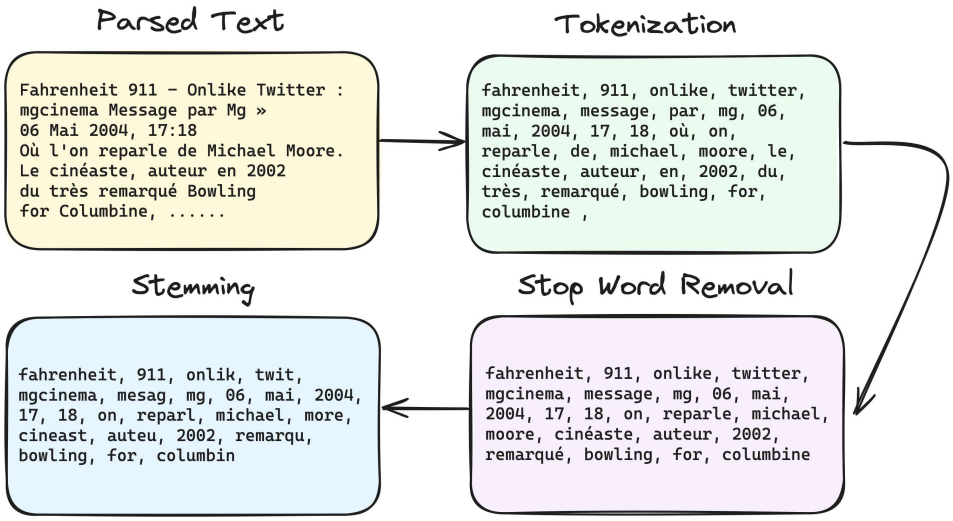


Figure 3: Analyzer Process.

Also, we used *FrenchElisionFilter* [11] that is addresses elision phenomena in French, where certain characters, such as 'l', 'd', 's', 't', 'n', and 'm', followed by an apostrophe, are contracted by eliminating the apostrophe and associated character.

3.3. Indexer

Creating an index is one of the main process where we generate a searchable database, known as an index, for parsed documents. This index holds crucial information about the documents such as the words and phrases they contain, their frequency, and their locations within the document. Indexing facilitates swift document retrieval by enabling users to search based on keywords or phrases. To accomplish this task, we developed the following components:

- **analyzer**: is the analyzer to be used, which is *LongEvalAnalyzer*.
- **similarity**: is an object needed to score the relevance of a document based on the query terms it contains. which can be implemented to be either Lucene default implementation that is based on a variant of the *Term Frequency-Inverse Document Frequency (TF-IDF)* model, or the modern alternative *BM25* which is used in our case.
- **ramBufferSizeMB**: the size in megabytes of the RAM buffer for indexing the documents.
- **indexPath**: the path to the directory where the generated index should be stored.
- **docsPath**: the path to the documents directory.
- **dpCls**: an object of the DocumentParser which is responsible for parsing the documents in the collection.

We used *BM25 Similarity* because unlike TF-IDF, where term frequency linearly affects the score, BM25 introduces a saturation point, which prevents the term frequency component from indefinitely influencing the score. BM25 scores a document based on the query terms appearing in it using the following formula:

$$\text{BM25}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)}$$

where D is the document being scored, Q is the query consisting of words $q_1, q_2, \ldots, q_n$, $f(q_i, D)$ is the frequency of the term q_i in document D, *IDF*(q_i) is the inverse document frequency of term q_i, $|D|$ is the length of the document, *avgdl* is the average document length in the text collection. Finally, k_1 and b are free parameters, we left $k_1 = 1.2$ and $b = 0.75$ same as most of the applications.

After setting the configuration of the indexer, it does the following:

- The indexer walks through the documents directory to find documents (specifically *.txt* files) and processes each file for indexing.
- It uses *DocumentParser* class, which is tasked with parsing the documents and creating structured data *ParsedDocument*, including documents identifiers and bodies.
- Each parsed document is converted into a Lucene Document object and added to the index. The document's ID and body are stored as fields within the Lucene Document.

The configurations used for the fields added to the Lucene Document are :

- **IDField**: The ID field is created by only storing the document ID without storing term frequencies or positions, which are unnecessary for unique identifiers like document IDs, so only the original value of the field (the document ID) is stored directly in the index, allowing it to be retrieved when querying the index.

- **BodyField**: The body field of the document is configured to store the terms resulted form splitting the body of the document into tokens, along with their frequencies, without storing the original text of the document in an attempt to reduce the size of the index.

3.4. Searcher

The Searcher's task is scanning indexed documents, analyzing user queries, and retrieving relevant information. It then presents a ranked list of documents that satisfy the user's information needs.

Our implementation does so by accepting the following parameters:

- **analyzer**: in this case, an instance of our Analyzer.
- **similarity**: we decided to use the BM25Similarity [12] function for the first stage of document retrieval due to its efficiency and higher effectiveness compared to other methods.
- **Run options**: parameters for the index path, the topics path, the run path and the run name, the number of the expected topics, and the maximum number of documents retrieved (in our case 1000).
- **Search options**: parameters for query boosting, query boosting value, number of documents to be re-ranked, score calculation mode, query expansion mode, and LLM used for generating the expansion.

3.4.1. Query Expansion

Query Expansion plays a valuable role in improving the performance of our Search Engines based on how it is used. We generate multiple expansions for each query by implementing a Python script. This script retrieves the **.trec* topic file and generates synonymous phrases using *Meta Llama 3* [13] and *Mistral-7B* [14] models, both of which are open-source. The prompt used for generating is as follows:

> Instruction: Please provide {num_expansion} synonyms in French for the given keyword that convey similar meanings, The output should be a list of words separated by commas without any further punctuation.
>
> The keyword is {word}.

Following this procedure, we slightly cleaned the generated file. Occasionally encountering low-quality or missing generations, we implemented a method in the search section to automatically switch to a second model if the initial one fails.

The sample result for the prompt is:

Table 1
Sample results for query expansion using the given prompt.

Query	Meta Llama 3	Mistral-7B
anti-virus gratuit	1. programmes antivirus libres 2. solutions antivirus gratuites.	1. logiciel antivirus gratuit 2. solution antivirus gratuit
bardeau	1. clôture 2. écran 3. barrage	1. plaque de bois 2. tableau 3. pannée

3.4.2. Query Boosting

Query Boosting is a technique used to adjust the score of documents retrieved by a search query, allowing for customization of document relevance based on specific criteria.

When executing a query, Lucene assigns a score to each matching document based on its relevance. Query boosting enables modification of these scores for particular documents or groups. Through experimentation, we found that mixing query expansion and boolean queries sometimes resulted in poorer outcomes. However, by introducing *Lucene's BoostQuery* [15], we observed improvements in our evaluation metrics. We experimented with three approaches:

- Multiplying by the number of expansions we have.
- Utilizing a fine-tuned parameter of 14.68, multiplied by the number of expansions we have.
- $14.68 \times (num\_expansion + \frac{total\_expansion}{num\_expansion} - 1)$ while
  - num_expansion is the number of queries we selected in query expansion.
  - total_expansion is the total number of queries we expanded.

  our reasoning for this approach was to prioritize exact terms when a word has a lot of meanings. However, our findings suggest that this strategy does not produce the anticipated results.

To summarize, we settled on the second approach, using a boolean query where we added at most three expansions with the *SHOULD* term, and boosted the main query with *MUST*.

3.4.3. Document Re-Ranking

Document Re-Ranking is the process of taking the initially ranked list of documents (or items) and re-evaluating their relevance or importance based on new information, constraints, or preferences. our approaches for document Re-Ranking are:

- **Secondary ranking function**: Apply a secondary ranking function that considers additional criteria or constraints.
- **Score adjustment**: Modify the scores of individual documents based on another score.

For the ranking function, we are considering two different approaches: using *Bidirectional Encoder Representations from Transformers (BERT)* models and using LLMs. In both cases, the objective is to compute the embeddings of the words and determine the *cosine similarity* between the query and the document.

After research, we attempt to find a fast *SBERT* [16] model and a well-tuned LLM to assess performance. We opted for *jina-reranker-v1-turbo-en*, designed for rapid reranking while maintaining competitive performance, leveraging JinaBERT [17] model as its foundation. Additionally, for the LLM, we chose *sentence-croissant-llm-base*, engineered to produce French text embeddings. It has been fine-tuned using the recently pre-trained LLM *croissantllm/CroissantLLMBase* [18].

In the end, we found that employing LLMs for re-ranking is computationally expensive and produces nearly identical results. Consequently, we decided to utilize *jina-reranker-v1-turbo-en*, ranking the first 200 documents and leaving the rest unchanged.

For score adjustment we used two ways:

- **Simple Mode**: change the score of the document directly based on secondary ranking function.
- **Harmonic Mode**: combine the BM25 score with the secondary ranking function score.

$$H = \frac{2}{1/x_1 + 1/x_2}$$

The *Harmonic* mode, based on the results, performs better.

4. Experimental Setup

The experimental setup for our *Information Retrieval (IR)* system includes using the LongEval collection, which is the official training collection for the 2024 LongEval IR Lab (https://clef-longeval.github.io/). The collection contains French-language web pages and queries, along with their English translations. We used the French data for our experiments.

To assess the performance of our IR system, we used the *trec_eval* executable to evaluate the results under various configurations. We monitored improvements in the following evaluation metrics produced by *trec_eval*:

- **num_ret**: Number of documents retrieved for each query.
- **num_rel**: Number of relevant documents for each query.
- **num_rel_ret**: Number of relevant documents retrieved for each query.
- **map**: Mean Average Precision, indicating the average relevance of retrieved documents across all queries.
- **rprec**: R-Precision, calculated at the rank corresponding to the number of relevant documents for each query.
- **p@5** & **p@10**: Precision at 5 and at 10, representing precision scores computed at the top 5 and 10 retrieved documents for each query.
- **nDCG**: Normalized Discounted Cumulative Gain, a metric evaluating ranked lists by considering item relevance.

Our project's Git repository is publicly available at (https://bitbucket.org/upd-dei-stud-prj/seupd2324-kalu/src/master), The code is openly accessible for replication. We used a MacBook Pro with an M2 Max chip, 12-core CPU, 30-core GPU, and 32GB RAM to compute our runs.

5. Results and Discussion

In this section, we present some of the most fitting results obtained during the development phase. We are considering five primary milestones that, after multiple trials, substantially improved our *Mean Average Precision (MAP)* score and the overall number of relevant documents retrieved. Several models were evaluated, focusing on re-ranking and query expansion techniques.

Table 2
Parameters used in the 5 different runs submitted to CLEF

Parameter	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**
Token Filter	Porter	FrenchLight	FrenchLight	FrenchLight	FrenchLight
Tokenizer	Standard	Standard	Standard	Standard	Standard
Length Filter	2-15	2-15	2-15	2-15	2-15
Stop Filter	"*None*"	"*stoplist-fr*"	"*stoplist-fr*"	"*stoplist-fr*"	"*stoplist-fr*"
Lower Case Filter	Yes	Yes	Yes	Yes	Yes
Similarity	BM25	BM25	BM25	BM25	BM25
Query Expansion	No	Yes	Yes	Yes	Yes
Query Expansion Model	-	Llama 3	Mistral*	Mistral*	Mistral*
Boolean Clause Main Query Mode	"*SHOULD*"	"*SHOULD*"	"*SHOULD*"	"*SHOULD*"	"*MUST*"
Re-ranking	No	Yes	Yes	Yes	Yes
Score Combination Mode	-	Simple	Simple	Harmonic	Harmonic
Num. of Re-rank Documnet	-	100	100	200	200

∗ If the model failed, it would switch to another one.

5.1. Results on training data

Table 3
Results for systems (top-1000 documents), on the French Train collection and the train query set of LongEval.

metrics	run1	run2	run3	run4	run5
num_q	597	599	599	599	597
num_ret	584902	598281	599000	599000	587943
num_rel	4344	4362	4362	4362	4350
num_rel_ret	3542	3553	3583	3583	3578
map	0.1853	0.2286	0.2313	0.2366	0.2374
gm_map	0.0484	0.0659	0.0744	0.0817	0.0841
Rprec	0.1756	0.2271	0.2281	0.23	0.2308
recal_10	0.231	0.2855	0.2857	0.2915	0.2925
recall_100	0.5666	0.574	0.5826	0.6174	0.6192
recall_1000	0.8123	0.812	0.821	0.821	0.8225
ndcg	0.3692	0.4065	0.4097	0.4161	0.4173
ndcg_rel	0.2875	0.3274	0.329	0.3344	0.3354
Rndcg	0.2243	0.269	0.2701	0.2745	0.2754
ndcg_cut_10	0.1954	0.2464	0.2458	0.2511	0.252
ndcg_cut_100	0.3162	0.3556	0.3591	0.3727	0.374
ndcg_cut_1000	0.3692	0.4065	0.4097	0.4161	0.4173
map_cut_10	0.1271	0.1691	0.1702	0.1729	0.1735
map_cut_100	0.1807	0.2244	0.2273	0.2331	0.2339
map_cut_1000	0.1853	0.2286	0.2313	0.2366	0.2374

Initially, we found out that using the *FrenchLightStemFilter* [19] as the stemmer, and adjusting the length filter from 2 to 15 (reflecting the tendency of French to have longer words), yielded very positive results [3]. Then to continue we introduce four models: base model, re-rank 100 documents with simple score combination mode, re-rank 100 documents with simple score combination mode using Mistral query expansion with a threshold of three words, and re-rank 100 documents with simple score combination mode using Llama query expansion with three words. The third model, utilizing Mistral query expansion, achieved the highest MAP of 0.044, surpassing the base model. Subsequently, two more models were introduced, we tried to see the difference between score combination models and handling empty expansion cases, so basically, we achieved a higher MAP of 0.0487 compared to the base model, handling empty cases if necessary, utilizing stopwords, and using harmonic score combination.

To sum up, The most successful approaches involved re-ranking using Mistral query expansion with Llama3 replacement, threshold three, and the inclusion of stopwords, with the harmonic mean performing the best among these methods.

Additional models were tested, including summarizing texts using LLM and integrating them into the original texts before indexing, or completely replacing the original text with the summary with *flan-t5-3b-summarizer*[20]. However, these approaches provided similar results to the simpler methods and required significantly more time to execute. Furthermore, various boosting methods were explored, but most decreased the MAP in the training dataset (see Section 3.4.2). There was also consideration of discarding the use of LLM for re-ranking due to its poor performance.

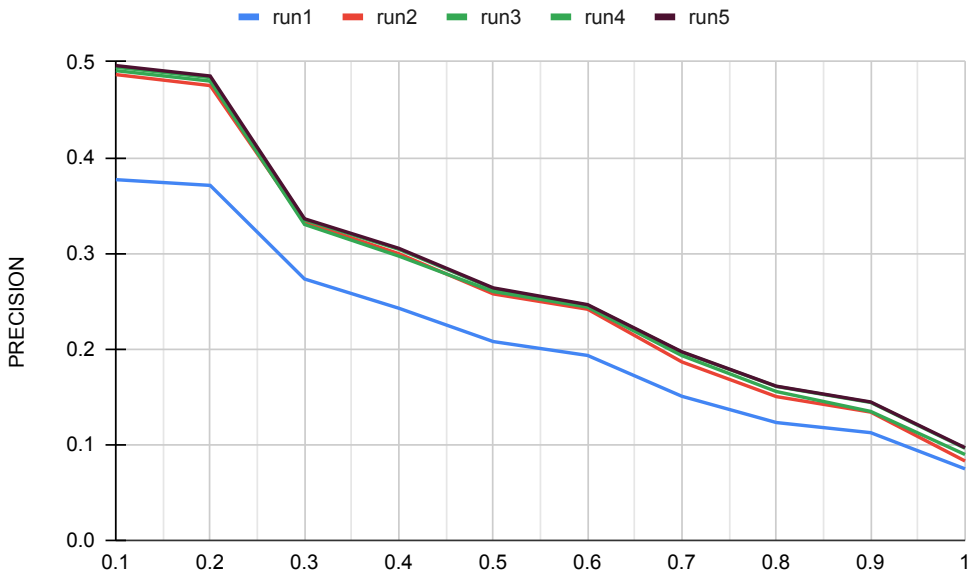


Figure 4: Standard Recall Levels vs Interpolated Precision

5.2. Results on Test data

In this section we have provided the results obtained by running our algorithms on each of the two available test collections which are short term and long term.

Table 4
Scores of all five systems on short term test collection.

Evaluation measure	run1	run2	run3	run4	run5
Map	0.1578	0.1855	0.1875	0.1922	0.1922
nDCG	0.2984	0.3225	0.3240	0.3302	0.3302
nDCG@10	0.1886	0.2247	0.2264	0.2297	0.2297
P@10	0.1472	0.1745	0.1752	0.1789	0.1789
Recall	0.5884	0.5867	0.5884	0.5884	0.5884

Table 5
Scores of all five systems on long term test collection.

Evaluation measure	run1	run2	run3	run4	run5
Map	0.1067	0.1400	0.1395	0.1430	0.1434
nDCG	0.2193	0.2502	0.2494	0.2535	0.2542
nDCG@10	0.1479	0.1921	0.1912	0.1931	0.1936
P@10	0.1145	0.1407	0.1397	0.1413	0.1417
Recall	0.4142	0.4136	0.4131	0.4131	0.4142

6. Statistical Analysis

In this section, we conduct a statistical analysis on the retrieval effectiveness for our five submitted runs to CLEF. This evaluation aims to assess each run's performance and find out how well the system retrieves and ranks relevant documents.

We compared the Normalized Discounted Cumulative Gain (nDCG) and Mean Average Precision (MAP) of each runs to understand the performance differences among them, considering both short-term and long-term evaluations. The analysis involves the use of tools such as box plots, two-way Anova and Tukey.

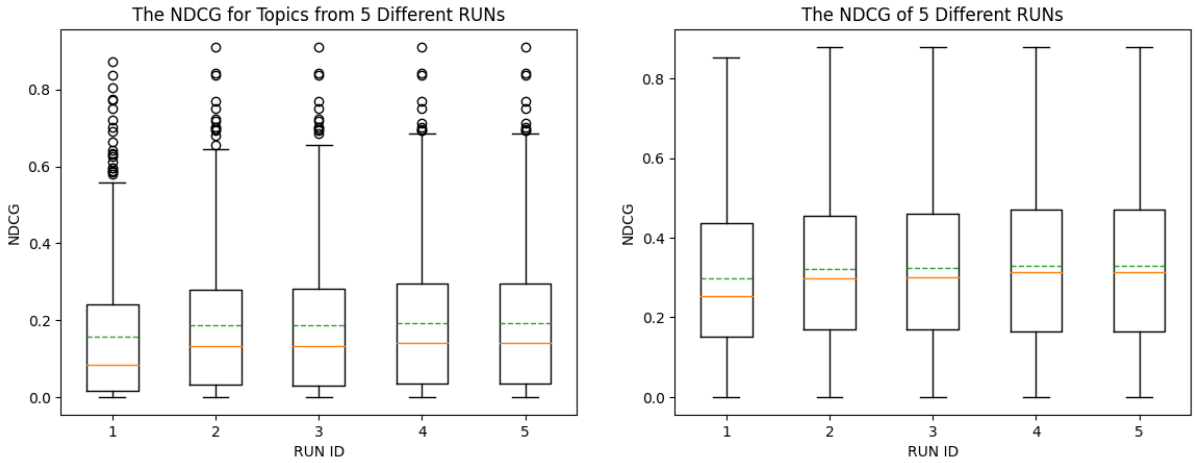
For analysis first we used box plots which are used to represent a distribution of data concisely. Additionally, we applied two-way Analysis of Variance (ANOVA) tests to explore the differences observed in both short-term and long-term evaluations. In addition, we use the Tukey Honest Significant Difference (HSD) test, a post-hoc analysis for ANOVA, which compares group means while controlling for multiple comparisons, to ensure reliable identification of significant differences.

Table 6
Recap of our runs submitted to CLEF.

runs	language	type
run1	French	Base
run2	French	Query Expansion using LLama 3
run3	French	Re-ranking simple mode
run4	French	Harmonic Re-ranking using Should
run5	French	Harmonic Re-ranking using Must

6.1. Box Plot

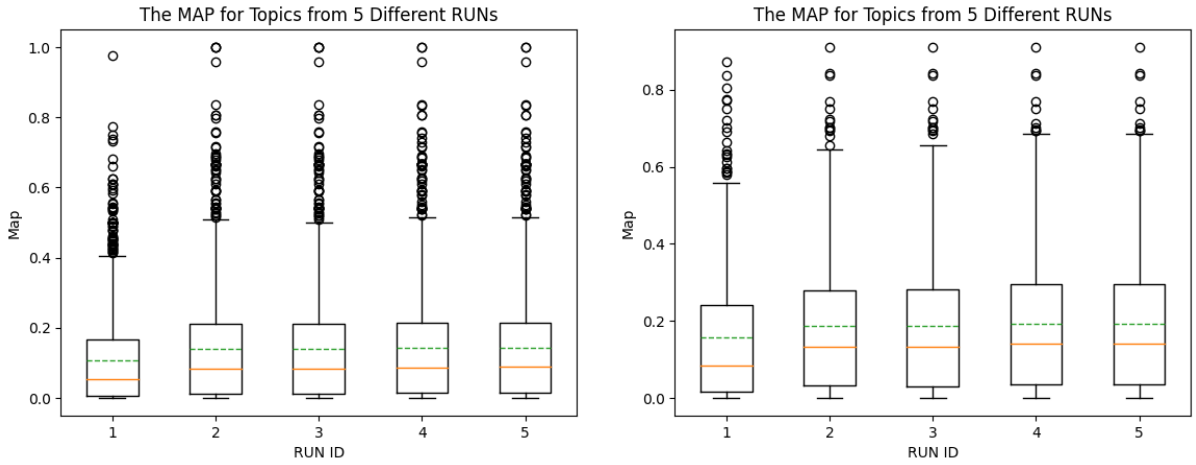
Box plots are graphical tools to represent a distribution of data concisely. In our case, we want to plot the distribution of the scores achieved by our submitted systems on each query of the different test sets, with respect to nDCG and Map.



(a) long-term set runs (b) short-term set runs

Figure 5: box plot for both the short-term and long-term set runs (nDCG performance)

By analysing the nDCG performance of all runs in short-term set we observe that run1 achieve lower nDCG scores, indicating their inferior effectiveness in capturing and ranking relevant documents while the other 4 runs exhibit approximately similar levels of performance. This nDCG performance result is also the same in long-term set runs.



(a) long-term set runs (b) short-term set runs

Figure 6: box plot for both the short-term and long-term set runs (Map performance)

From the boxplot, we can observe the distribution of MAP scores for each run. By analysing the Map performance of all runs in short-term and long-term set we observe that run1 has the lowest Map scores, indicating their inferior effectiveness in accuracy, run2 and run3 have approximately same Map scores in short and long-term evaluations while run4 and run5 have the highest Map scores with a slight difference with run2 and run3.

6.2. Two-way ANOVA

In a two-way ANOVA test, we check if the factors Topic and System can influence the results and we test on both MAP and nDCG measures.

Table 7
two way ANOVA Results for short-term nDCG

Source	df	SS	MS	F	PR(>F)
Columns(Systems)	4.0	0.2785	0.0696	21.3414	3.594524e-17
Rows(Topics)	403.0	82.8188	0.2055	62.9957	0
Error	1612.0	5.2587	0.0033	-	-
Total	2019.0	88.3559	-	-	-

Table 8
two way ANOVA Results for short-term Map

Source	df	SS	MS	F	PR(>F)
Columns(Systems)	4.0	0.3358	0.0840	24.6253	8.216775e-20
Rows(Topics)	403.0	67.4513	0.1674	49.0915	0
Error	1612.0	5.4960	0.0034	-	-
Total	2019.0	73.2831	-	-	-

Table 9
two way ANOVA Results for long-term nDCG

Source	df	SS	MS	F	PR(>F)
Columns(Systems)	4.0	1.2535	0.3134	11.8346	1.4110e-09
Rows(Topics)	1511.0	103.3381	0.0684	2.5829	1.8578e-142
Error	6044.0	160.0363	0.0265	-	-
Total	7559.0	264.6278	-	-	-

Table 10
two way ANOVA Results for long-term Map

Source	df	SS	MS	F	PR(>F)
Columns(Systems)	4.0	1.4099	0.3525	19.2277	9.8898e-16
Rows(Topics)	1511.0	71.7684	0.0475	2.5909	1.9168e-143
Error	6044.0	110.7997	0.0183	-	-
Total	7559.0	183.9780	-	-	-

From the result of the two-way ANOVA test we can conclude that both factors (System and Topic) are important in influencing the performance measures (nDCG and MAP) in both short-term and long-term evaluations.

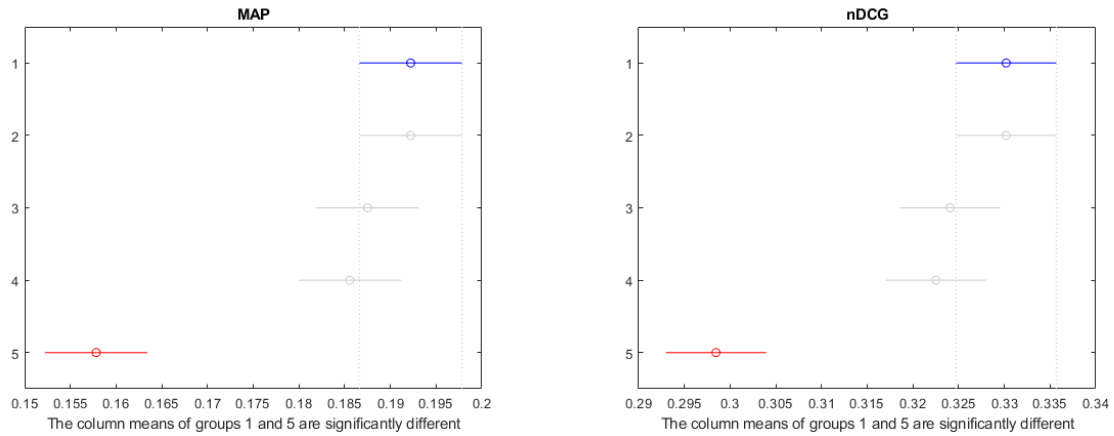


Figure 7: Tukey's HSD test for all the five runs in long-term evaluation

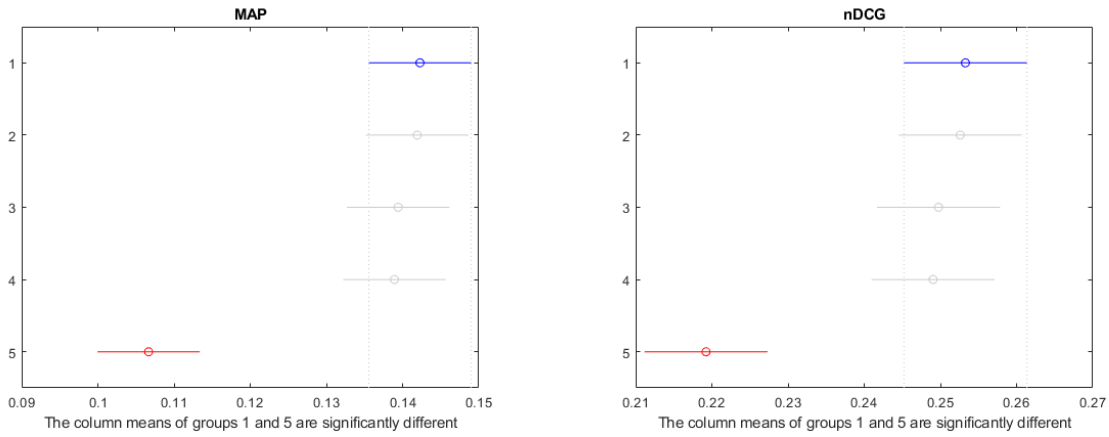


Figure 8: Tukey's HSD test for all the five runs in long-term evaluation

7. Conclusions and Future Work

In this work, we present our approach to the CLEF Long Eval LAB 2024 task, which aimed to develop an effective and efficient search engine for web documents. Our approach consisted of combining different techniques, including query expansion, re-ranking, and the use of large language models for different purposes. Our experiments showed good results for our approach, with better effectiveness and efficiency than the baseline system provided by CLEF. Combining two scores in the re-ranking phase also improved retrieval performance. We found several areas to improve our approach further.

One promising direction is to use text summarization and title extraction techniques in the parsing part. While we experimented with this approach, it didn't generate significant improvements due to efficiency concerns. However, we believe that refining this technique or exploring alternative approaches could lead to better results.

Another idea is to embed documents using different methods [21] for re-ranking, or text chunks and their summaries because chunking text documents into small pieces is an interesting technique that increases the accuracy and quality of the system which could help capture nuanced semantic relationships between documents. Additionally, including context-awareness when calling LLMs to generate synonyms might have a positive impact on the overall retrieval performance.

Furthermore, fine-tune our re-ranker with SBERT using training data and implement a custom re-ranker specific to this specific task. By leveraging the strengths of different models and techniques, we hope to achieve even better results and push the boundaries of what is possible in LongEval information retrieval.

References

[1] Qwant, About qwant, https://about.qwant.com/en/, 2023. Accessed: 2023-05-20.

[2] P. G. R. Deveaud, G. Gonzalez-Saez, P. Mulhem, L. Goeuriot, F. Piroi, M. Popel, Longeval-retrieval: French-english dynamic test collection for continuous web search evaluation, 2023. arXiv:2303.03229.

[3] G. Antolini, N. Boscolo, M. Cazzaro, M. Martinelli, S. Safavi, F. Shami, N. Ferro, et al., Seupd@ clef: Team close on temporal persistence of ir systems' performance, in: CEUR WORKSHOP PROCEEDINGS, volume 3497, CEUR-WS, 2023, pp. 2368–2395.

[4] E. Bolzonello, C. Marchiori, D. Moschetta, R. Trevisiol, F. Zanini, N. Ferro, et al., Seupd@ clef:

Team faderic on a query expansion and reranking approach for the longeval task, in: CEUR WORKSHOP PROCEEDINGS, volume 3497, CEUR-WS, 2023, pp. 2252–2280.
[5] M. Günther, J. Ong, I. Mohr, A. Abdessalem, T. Abel, M. K. Akram, S. Guzman, G. Mastrapas, S. Sturua, B. Wang, M. Werk, N. Wang, H. Xiao, Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2024. arXiv:2310.19923.
[6] A. Lucene, Standardtokenizer, https://lucene.apache.org/core/6_6_0/core/org/apache/lucene/analysis/standard/StandardTokenizer.html, 2024. Accessed: 2024-05-20.
[7] A. Lucene, Frenchlightstemmer, https://lucene.apache.org/core/6_2_0/analyzers-common/org/apache/lucene/analysis/fr/FrenchLightStemmer.html, 2024. Accessed: 2024-04-20.
[8] A. Lucene, Luke, https://lucene.apache.org/core/8_11_0/luke/index.html, 2024. Accessed: 2024-05-20.
[9] A. Lucene, Apache lucene, https://lucene.apache.org/, 2023. Accessed: 2023-05-20.
[10] Kaggle, Frenchkagglestoplist, https://www.kaggle.com/datasets/heeraldedhia/stop-words-in-28-languages?select=french.txt, ????
[11] A. Lucene, Lucene elisionfilter, https://lucene.apache.org/core/7_3_1/analyzers-common/org/apache/lucene/analysis/util/ElisionFilter.html, 2024. Accessed: 2023-04-20.
[12] A. Lucene, Lucene bm25similarity, https://lucene.apache.org/core/7_0_1/core/org/apache/lucene/search/similarities/BM25Similarity.html, 2024. Accessed: 2024-04-20.
[13] AI@Meta, Llama 3 model card (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
[14] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7b, 2023. arXiv:2310.06825.
[15] A. Lucene, Lucene boostquery, https://lucene.apache.org/core/7_3_1/core/org/apache/lucene/search/BoostQuery.html, 2024. Accessed: 2024-04-20.
[16] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: https://arxiv.org/abs/1908.10084.
[17] M. Günther, J. Ong, I. Mohr, A. Abdessalem, T. Abel, M. K. Akram, S. Guzman, G. Mastrapas, S. Sturua, B. Wang, M. Werk, N. Wang, H. Xiao, Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2023. arXiv:2310.19923.
[18] M. Faysse, P. Fernandes, N. M. Guerreiro, A. Loison, D. M. Alves, C. Corro, N. Boizard, J. Alves, R. Rei, P. H. Martins, A. B. Casademunt, F. Yvon, A. F. T. Martins, G. Viaud, C. Hudelot, P. Colombo, Croissantllm: A truly bilingual french-english language model, 2024. arXiv:2402.00786.
[19] A. S. Foundation, Apache solr frenchlightstemfilter, https://solr.apache.org/guide/6_6/language-analysis.html#LanguageAnalysis-FrenchLightStemFilter, 2024. Accessed: 2024-04-20.
[20] J. Clive, Multi-purpose summarizer (fine-tuned google/flan-t5-xl on several summarization datasets), https://huggingface.co/jordiclive/flan-t5-3b-summarizer, 2023. URL: https://huggingface.co/jordiclive/flan-t5-3b-summarizer, apache 2.0 and BSD-3-Clause License. Fine-tuned on various summarization datasets including xsum, wikihow, cnn_dailymail/3.0.0, samsum, scitldr/AIC, billsum, TLDR. Designed for academic and general usage with control over summary type by varying the instruction prepended to the source document.
[21] N. Muennighoff, N. Tazi, L. Magne, N. Reimers, Mteb: Massive text embedding benchmark, arXiv preprint arXiv:2210.07316 (2022). URL: https://arxiv.org/abs/2210.07316. doi:10.48550/ARXIV.2210.07316.