SEUPD@CLEF: Team MOUSE on Enhancing Search Engines Effectiveness with Large Language Models.

Notebook for the LongEval Lab at CLEF 2024

Lorenzo Cazzador[1], Francesco Luigi De Faveri[1], Filippo Franceschini[1], Lorenzo Pamio[1], Samuel Piron[1] and Nicola Ferro[1]

[1]*University of Padua, Italy*

Abstract

The deterioration of the performances of *Information Retrieval Systems (IRSs)* over time remains an open issue among the *Information Retrieval (IR)* community. With this study for Task 1 of the *Longitudinal Evaluation of Model Performance LAB (LongEval)* at *Conference and Labs of the Evaluation Forum (CLEF)* 2024, we aim to propose and analyze the performance of an IRS that is able to handle changes over time in the data. In addition, the model uses different *Large Language Models (LLMs)* to enhance the effectiveness of the retrieval process by rephrasing the queries for the search and the reranking of the retrieved documents. With an in-depth analysis of the performance of the MOUSE group Retrieval System, using the datasets provided by the organisers of CLEF, the proposed model was able to reach a *Mean Average Precision (MAP)* of 0.22 and a *Normalized Discounted Cumulated Gain (nDCG)* of 0.40 for the English collection, increasing the performance for the original French collection up to 0.31 and 0.50, for MAP and nDCG respectively.

Keywords

Information Retrieval, Search Engines, Query Expansion, Reranking, Large Language Models

1. Introduction

Search Engines (SEs) have become an integral tool for people in retrieving information throughout their daily routines. Recent research studies [1, 2] in the field of *Information Retrieval (IR)* have analyzed the problem of evaluating the persistence of the *Information Retrieval Systems (IRSs)* performances over time, observing an important problem: the effectiveness of the system deteriorates due to changes in the data used for the searching of documents. Therefore, the *Conference ad Labs of the Evaluation Forum* (CLEF) 2024 LongEval task proposed the implementation of a SE able to handle changes in data over time. The dataset provided was obtained by scraping the *Qwant* [3] SE to acquire a large collection of information comprising a corpus of web pages, queries and user interactions. Such a dataset was collected over several months to reflect the changes in the search performances of the users. In addition, the documents in the collection were then selected to be able to well evaluate retrieval on the queries at the time they were collected, thus representing a good simulation for the changes over time. The dataset was provided in the original language of the collection, French, and these documents were automatically translated into English.

This work reports the proposed solution implemented by the MOUSE group at the University of Padua as part of the Course of Search Engines. Alongside the traditional search pipeline, the MOUSE system also utilizes query expansion and reranking techniques based on the text generated by *Large Language Models (LLMs)* to enhance the retrieval capability of the queries used. Finally, the best model proposed was able to achieve a MAP and nDCG of 0.22 and 0.40 for the English collection, respectively, increasing the performances to 0.31 and 0.50 on the original French data.

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

lorenzo.cazzador.1@studenti.unipd.it (L. Cazzador); francescoluigi.defaveri@phd.unipd.it (F. L. De Faveri); filippo.franceschini.3@studenti.unipd.it (F. Franceschini); lorenzo.pamio@studenti.unipd.it (L. Pamio); samuel.piron@studenti.unipd.it (S. Piron); nicola.ferro@unipd.it (N. Ferro)
https://www.dei.unipd.it/~defaverifr/ (F. L. De Faveri); https://www.dei.unipd.it/~ferro/ (N. Ferro)
0009-0005-8968-9485 (F. L. De Faveri); 0000-0001-9219-6239 (N. Ferro)

The paper is organized as follows: Section 2 briefly introduces some related works for past LongEval tasks at CLEF 23; Section 3 describes our approach; Section 4 explains our experimental setup; Section 5 discusses our main findings; finally, Section 6 draws some conclusions and outlooks for future work.

2. Related Works

Section 2 describes related works used during the implementation of the MOUSE system. Subsection 2.1 proposes a walk-through of two of the studies that have faced the problem of Longitudinal Evaluation of IRSs at the CLEF 23 LongEval Laboratory.

The implementation of our IRS uses as base code the one proposed during the Search Engines course in the academic year 2023/24. The classes implemented in Java provided during the course were *ParsedDocument, DocumentParser, TipsterParser, DirectoryIndexer, BodyField, TopicsReader, Searcher*, and they were used for the Tipster collection [4]. Our implementation uses these classes as a starting point and is described in depth in Section 3.

2.1. Past LongEval research

Prior research in the field of IR has explored different methods for evaluating the performance of an IRS considering changes over time. In their study, Antolini et al. [5] presented a pipeline for document processing that involves the elimination of irrelevant scripts and advertisements, followed by an analysis utilizing various techniques such as stemming and stopword filtering. After that, the documents are then indexed for efficient retrieval. As in our study, the queries are expanded using the ChatGPT 3.5 Turbo [6] model to enhance the performance before the searching step. We assumed a similar approach, changing the model used for the query expansion process, adopting Open Source models with higher performance if compared to the ChatGPT 3.5 Turbo[1].

Differently, Bolzonello et al. [7] used pre-trained *Language Model (LM)* to rerank the results obtained by searching documents with BM25. The authors used a T5-base [8] and a Bert-base [9] models, achieving an improvement of 3.5% and 8.5% on MAP and nDCG respectively. Through the inclusion of LLMs in our query expansion process, we were able to top the results of Bolzonello et al. [7] without requiring any additional reranking.

3. Methodology

In this section of the paper, we describe the methodology adopted, with a focus on the architecture of our system, implemented using the Apache Java Lucene library along with other Python code for the query expansion step. Figure 1 provides the traditional Apache Lucene system.

We also provide a general overview of the workflow implemented, Figure 2. We provide a brief explanation of the main phases of the process:

- **Parsing** and **Analyzing**: the text sanitization of queries and documents conducted to extract relevant information and remove useless noise. The raw input query and documents are tokenized, stemmed and filtered in order to remove useless code, HTML tags and special characters, such as emoji.
- **Indexing**: each document undergoes an indexing process, retaining only essential information. Indexed documents contain an ID field with the document's identifier in the collection and a content field comprising the entire body of the document.
- **Query Expansion**: reformulating the queries with a LLM to broaden the scope of the user query. By supplementing original query terms with additional and contextually relevant terms, the effectiveness of the topic retrieval capability is increased.

[1]https://artificialanalysis.ai/

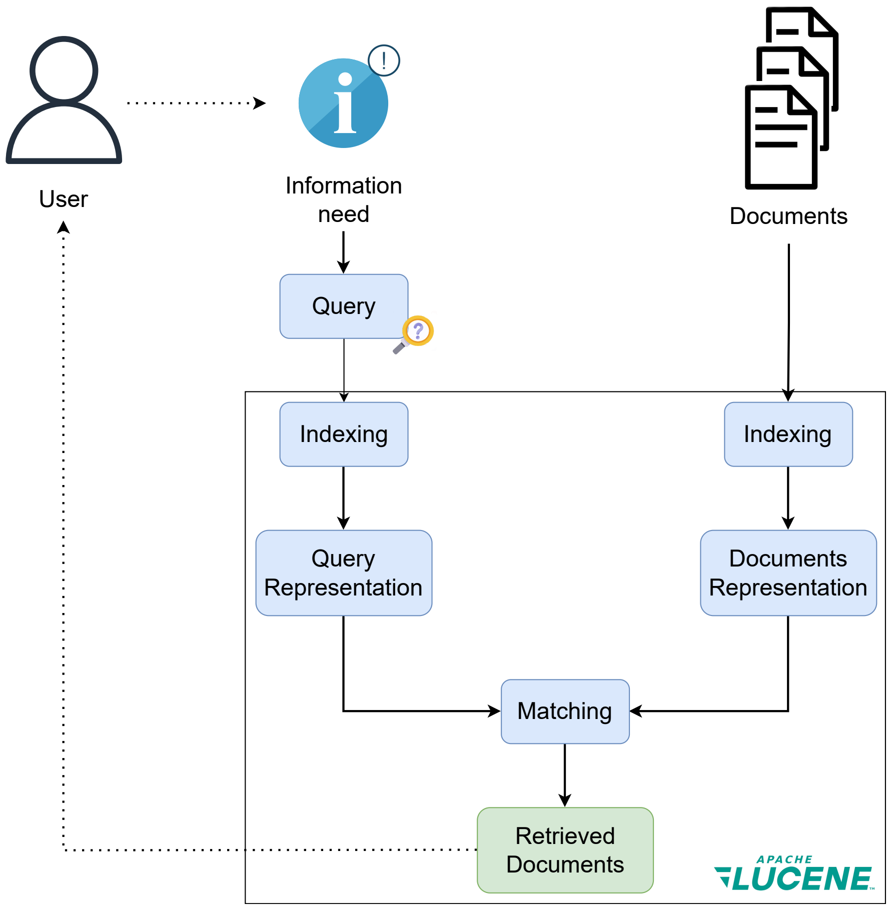


Figure 1: Traditional Y model for Search Engines using the Apache Lucene Java library. The blue boxes represent the different steps and objects used in the retrieval, while the green box stands for the final result, i.e., the list of retrieved documents.

- **Reranking**: sorting the obtained retrieved document list considering new scores provided by a Transformer based model. By rearranging the retrieved documents, the reranking phase enhances the user's search, presenting more relevant results at the top of the list.

3.1. Parser

Firstly, we downloaded the data from the LongEval data website[2]. Then, before starting the system implementation, we inspected the documents provided by CLEF LongEval 2024 organizers. During this initial phase, we conducted a thorough analysis of the data to gain a deeper understanding of how to optimize the parsing of documents and queries. This was a critical step in ensuring the effectiveness of the search processes and improving the overall performance.

The first phase of the workflow consisted of parsing the documents present in the collection; at the end of the parsing process, unnecessary noise is removed from each document. Three main Java classes were implemented to help this process:

- **DocumentParser**: This Java class is helpful in reading different types of documents and working with their content.
- **MouseParserJson**: This Java class is a parser for documents provided in JSON format for those supplied by CLEF LongEval 2024 organizers.

[2]https://clef-longeval.github.io/data/

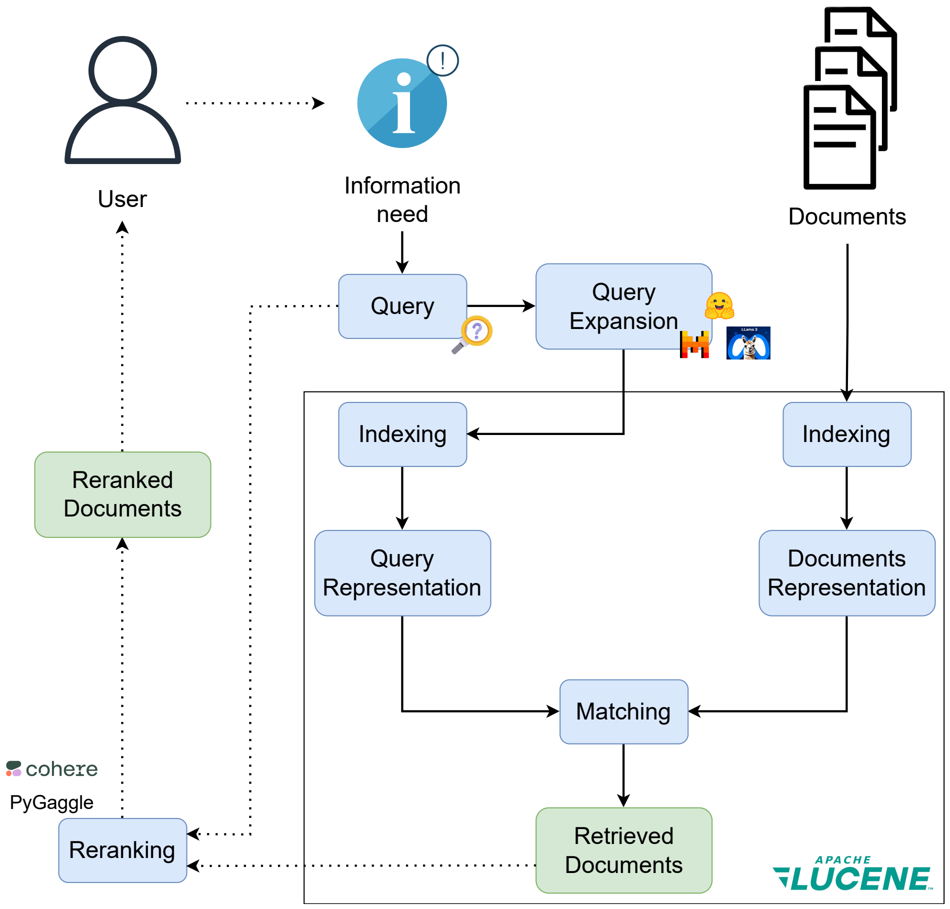


Figure 2: The MOUSE pipeline implemented in this study. The blue boxes represent the different steps and objects used in the retrieval, while the green box stands for the final result, i.e., the list of retrieved and reranked documents. In the diagram, next to the query expansion and reranking phase, the icons of the LLMs are shown.

- **ParsedDocument**: This Java class represents the parsed document ready to be indexed. The class collects two fields: the id and the body, i.e., the document's content comprising the text to be parsed.

Following implementation, we executed the aforementioned classes and stored the resulting data for the subsequent retrieval phase, which encompasses the analyzer step. During this phase, the *MouseParserJson* class was utilized to iterate over the input documents, while the *ParsedDocuments* class was employed to represent the document that required indexing.

3.2. Analyzer

After the parsing phase, we want to process and manipulate texts that come from queries and documents. We first decided to test the default Lucene Analyzer[3], and then we implemented a custom version of it: the *MouseAnalyzer*. This class has been used to manipulate words before the retrieval process.

Before explaining the analyzer, we need to explain the class *MouseParams* used by the Analyzer briefly. Such a class helps initialize the parameters the analyzer uses. Furthermore, we defined a *TokenizerType* attribute to specify which kind of tokenizer the system should use:

- **Whitespace**: the *WhitespaceTokenizer* is a tokenizer from the Lucene Library[4] that divides text at whitespace characters as defined by the method *isWhitespace*[5].

[3]https://lucene.apache.org/core/9_10_0/core/org/apache/lucene/analysis/Analyzer.html
[4]https://lucene.apache.org/core/9_10_0/analysis/common/org/apache/lucene/analysis/core/WhitespaceTokenizer.html
[5]https://docs.oracle.com/javase/8/docs/api/java/lang/Character.html?is-external=true#isWhitespace-int-

- **Letter**: the *LetterTokenizer* is a tokenizer from the Lucene Library[6] that divides text at non-letters.
- **Standard**: the *StandardTokenizer* is a tokenizer that divides text according to the Word Break rules from the Unicode Text Segmentation algorithm [10], which considers numbers and acronyms as separate tokens.

The file also contains a *StemFilterType* attribute, which represents the Stemming filter to use, with possible values comprising the following typology:

- **EnglishMinimal**: A minimal stemmer for English considering only plural words[7].
- **Porter**: An implementation of the Porter stemming Algorithm [11].
- **K**: An implementation of the Krovetz stemming Algorithm [12], by using the native class KStem-Filter in the Lucene library[8].
- **Lovins**: An implementation of the Lovins stemming Algorithm [13].
- **SnowBall**: An implementation of the Snowball stemmer, a native class in the Lucene library[9].
- **French**: An implementation of the "UniNE" algorithm [14].
- **None**: This flag says that no stemmer should be used.

CLEF LongEval 2024 organizers provided a collection of documents in English and French. We chose the best parameters for each language in our system based on the different trial run results obtained. The French documents were processed with:

- **Tokenizing**: the *StandardTokenizer* had the best overall performance.
- **Stemming**: the *StemFilterType* used in the system was the *French* one.
- **Stopword Removal**: the *StopFilter* is used to remove common and frequent words (e.g. "le", "et", and "à") from the text. In our system, we tried some stoplists with different lengths and words; the best we found was *stopwords-fr.txt* and *train24-top125-nominmaxlen.txt.* The first list of stopwords was provided by the Open Source project for stopwords removal [15]; the second list was computed by obtaining the first 125 terms with the higher frequency in the collection analyzing the index of the documents without applying any stemming or a stoplist. All the lists are available in the MOUSE repository[10].
- **Elision Removal**: the *ElisionFilter* was implemented in this system. Elisions in French, such as the contractions found in phrases (e.g. "aujourd'hui" or "qu'il"), were identified and handled appropriately to maintain the integrity of words in the analysis.

For the English documents, instead, were used:

- **Tokenizing**: the *StandardTokenizer* had the best overall performance.
- **Stemming**: the *StemFilterType* used in the system was the *Porter* one.
- **Stopword Removal**: the *StopFilter* is used to remove common and frequent words (e.g. "the", "an", and "that") from the text. In this case, we tested different stoplists, and as the final lists we decided to use for the final runs the *stopwords-en.txt* and *train24-top125-nominmaxlen.txt.*
- **Possessive Removal**: the *EnglishPossessiveFilter* was implemented in this system. It removes the possessives ('s) from words.

3.3. Searcher

The searching procedure starts when a user submits a query to the system, which then analyzes the query and searches through indexed documents to find relevant information. The system aims to retrieve and rank documents that align with the user's query, returning a list of results that best match the user's information needs, ordered by relevance. In this section, we explain some searching techniques that we implemented in our methodology.

[6] https://lucene.apache.org/core/9_10_0/analysis/common/org/apache/lucene/analysis/core/LetterTokenizer.html
[7] https://lucene.apache.org/core/9_10_0/analysis/common/org/apache/lucene/analysis/en/EnglishMinimalStemmer.html
[8] https://lucene.apache.org/core/9_10_0/analysis/common/org/apache/lucene/analysis/en/KStemFilter.html
[9] https://lucene.apache.org/core/9_10_0/analysis/common/org/tartarus/snowball/SnowballStemmer.html
[10] https://bitbucket.org/upd-dei-stud-prj/seupd2324-mouse/src/master/

3.3.1. BM25

For evaluating the matching between parsed and analyzed query $Q = (q_1, \ldots, q_n)$ and document D, we used the Okapi BM25 [16, 17] scoring function. In Equation 1 the scoring function is reported, where $f(q_i, D)$ is the frequency that a query term q_i appears in the document D, IDF is the *Inverse Document Frequency (IDF)* of the query terms [18], avgdl is the average document length in the text collection from which documents are taken from. In implementing our system, we employed the Okapi BM25 retrieval system offered by Lucene[11]. Moreover, we used the default settings provided in the Lucene library, i.e., $k = 1.2$ and $b = 0.75$. Accordingly, we found that the BM25 scoring function performed effectively without tuning these default parameters.

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

3.3.2. Fuzzy

A fuzzy query is a type of search query that finds matches even when the search terms do not exactly match the terms in the documents. This helpful approach is used to get a query's (partial) match to a document, even if there are some variations, e.g., words misspelt, abbreviations and typographical errors. Fuzzy is a powerful Lucene Library[12] established on the similarity between terms: such similarity measurement is based on the Damerau-Levenshtein algorithm [19], which is a method for computing the closeness between two strings, which takes into account the number of insertion, deletion, substitution, and transposition operations needed to transform one string into the other.

3.3.3. Spell-Checker

During the first analysis of the queries collection, we noticed that some of them contained words with grammatical errors:

Wrong Queries:

- *emploi terriotrial*
- *espace sheingen*

We implemented the Spell-Checker Lucene Library[13] to fix grammatical errors in the queries. Such a method is used inside the Searcher class, creating when the system encounters a token that is not present in the Indexer. We inspected the dictionary provided to see if the wrong word could be replaced with a correct version of it (or with a similar one). Thus, the queries become:

Corrected Queries:

- *emploi territorial*
- *espace schengen*

3.3.4. Word N-Grams

The Searcher, 3.3, also implements the *ShingleFilter*, a special filter that constructs shingles, i.e., n-grams tokens from a given stream of terms. They are a special sentence technique analysis that divides words of a sentence into sequences of *n* consecutive words; they improve search relevance.

For example, if we have a sentence like "Let's try it" we can have "let's try, try it, let's it" and we see that the sentence was divided into three shingles. We handled the base case in which the query has only

[11]https://lucene.apache.org/core/9_10_0/core/org/apache/lucene/search/similarities/BM25Similarity.html
[12]https://lucene.apache.org/core/9_10_0/core/org/apache/lucene/search/FuzzyQuery.html
[13]https://lucene.apache.org/core/9_10_0/suggest/org/apache/lucene/search/spell/SpellChecker.html

one word, and then the shingle is the query itself. Specifically, we set two variables: the *maxShingleSize* represents the maximum number of shingles a query should be divided into, and *shingleProximity* that is a measure of the distance of terms in shingles used to identify documents in which the search term appears. We tested with and without this filter and saw a remarkable increment in the MAP. The Word N-grams method is applied for queries that contain more than a single word, creating overlapping sequences of terms.

3.4. Query Expansion

To improve the effectiveness of the queries, we adopted a query boosting methodology based on LLM, as previously done by [5]. Figure 3 represents the pipeline used.

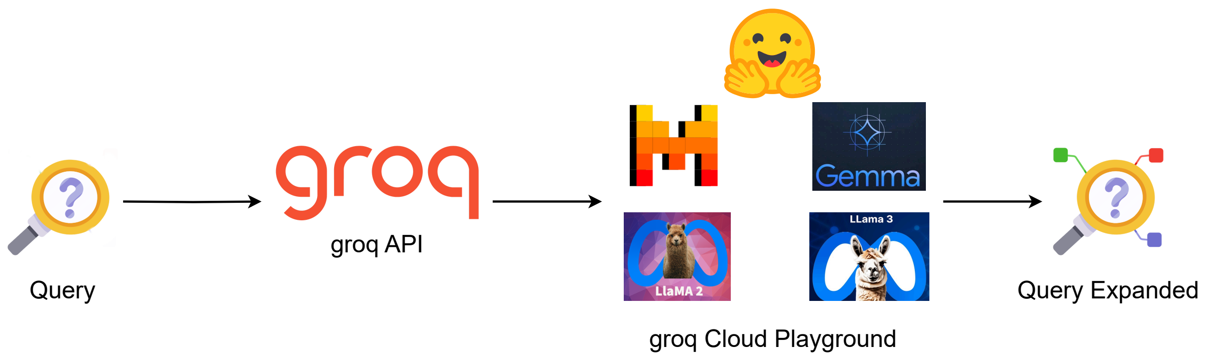


Figure 3: Query Expansion Pipeline.

We wrote the python script `queryExpansion.py` to interact with the groq API[14] and the models offered by the organization in their cloud infrastructure, i.e., LLaMA [20] models (LLaMA3 7b and LLaMA2 70b), Mixtral 8x7b [21], and Gemma 7b [22], deployed using the version provided by the Hugging Face platform[15]. To have access to the models, we generated a `GROQ_API_KEY` in the groq cloud platform and saved it as a configuration file to be accessed by the script. Since we used the Free Beta program of the service, we worked under some limitations: the number of possible requests per minute is limited to 30, and we could not fine-tune any model. Nevertheless, to exploit the capabilities of the LLMs used, we tried different contextualizations in the prompt submitted, identifying as most effective the following, written in French (translating the English original prompt with the help of the Google Translation service).

Prompt:

*"Extend the following query with 20 related terms or expressions in French: **QUERY**. Only print terms on a single line, separated by commas, do not add additional text or explanations."*

French translation:

*"Étendre la requête suivante avec 20 termes ou expressions liés en French: **QUERY**. Imprimez uniquement les termes sur une seule ligne, séparés par des virgules, n'ajoutez pas de texte ou d'explications supplémentaires."*

Hence, the script `queryExpansion.py`, takes as input the path to the query dataset path, passed with the "`-d`" flag, and the model name "`-m`" to be used, storing the final expanded queries in a .tsv file to be used for the retrieval phase. The communication between the localhost and the model used in the cloud respects the API limit by resting 60 seconds every 30 queries submitted in order to avoid the "`429 TooManyRequests`" HTTP Error. The results are saved in the predefined directory, storing the query id, original query, expanded query, and model used for the expansion.

[14]https://github.com/groq/groq-python
[15]https://huggingface.co/

3.5. Reranking

Figure 4 shows a schematic view of the reranking performed to rearrange the results of the first retrieval. For such a task, we tried three different approaches: the first was provided by connecting with the Cohere API[16] to use LLMs for performing the reranking scoring computation. Then, we used the Pygaggle Library[17] for interacting with deep neural architectures for text ranking, designed in Python. We also tried the rerank model *ms-marco-MultiBERT-L-12*, with the Multi-lingual support provided by the Python module FlashRank[18]. Finally, we found that the first two approaches gave us the best performances and decided to discard the third one to avoid encumbering.

Furthermore, the Cohere approach, thanks to the API service, gives us the opportunity to rerank up to 100 documents; on the other hand, Pygaggle runs the model on the local machine and, with our computational power, we were able to rerank approximately 20 documents per run. We wrote a simple Python script for each of these approaches with two functions: *load_ranker* and *rerank*. The former is called inside the constructor of the *Searcher* class, loading the model and saving it into a global variable. After that, the method *rerank* is called in the searching process after retrieving documents. This method receives the query title and the contents of documents as parameters to perform the reranking.

Once the scores have been computed, we used the normalization function proposed by Bolzonello et al. [7], Equation 2 to create the final ranking.

$$\text{nScore}_{rr}(i) = \left(\text{Score}_{rr}(i) + \min_{j \in [1,n]} \text{Score}(j)\right) \cdot \frac{\text{Score}_{\text{BM25}}(1)}{\text{Score}_{rr}(1)} \tag{2}$$

where $\text{Score}_{\text{BM25}}(i)$ is the score given by BM25 for the document at rank position i and $\text{Score}_{rr}(i)$ is the score computed by the reranker for the document at rank position i, while n is the total number of documents reranked. The ratio $\frac{\text{Score}_{\text{BM25}}(1)}{\text{Score}_{rr}(1)}$ preserves the score computed by the first retrieved document from the BM25 IRS. Finally, as done by Bolzonello et al. [7], the final score is computed, Equation 3.

$$\text{finalScore}(i) = \text{mntr} + (1-\alpha) \cdot \text{Score}_{BM25}(i) + \alpha \cdot \text{nScore}_{rr}(i) \tag{3}$$

where mntr is the maximum score of docs that are not reranked, preserving the order of the leftover docs. For the experiments, we used $\alpha = 0.6$.

The data presented in Table 1 outlines the models utilized for query expansion and reranking, as well as other information on the runs submitted to CLEF24 LongEval.

Table 1
Summary of parameters for the runs submitted to CLEF 24 LongEval.

English					
Parameter	*Run 1*	*Run 2*	*Run 3*	*Run 4*	*Run 5*
Stem Filter	Porter	Porter	Porter	Porter	Porter
Tokenizer	Standard	Standard	Standard	Standard	Standard
Stoplist	train24-top125-nominmaxlen.txt	train24-top125-nominmaxlen.txt	train24-top125-nominmaxlen.txt	stopwords-en.txt	-
Query Expansion Model	Llama3-70b	Llama3-70b	Mixtral-8x7b	Llama3-70b	Mixtral-8x7b
Reranking Model	Pygaggle-Luyu-20-w06	Cohere-100-w06	Pygaggle-Luyu-20-w06	-	-

French					
Parameter	*Run 6*	*Run 7*	*Run 8*	*Run 9*	*Run 10*
Stem Filter	French-Light	French-Light	French-Light	French-Light	French-Light
Tokenizer	Standard	Standard	Standard	Standard	Standard
Stoplist	train24-top125-nominmaxlen.txt	train24-top125-nominmaxlen.txt	train24-top125-nominmaxlen.txt	stopwords-fr.txt	-
Query Expansion Model	Llama3-70b	Llama3-70b	Mixtral-8x7b	Llama3-70b	Mixtral-8x7b
Reranking Model	Pygaggle-Luyu-20-w06	Cohere-100-w06	Pygaggle-Luyu-20-w06	-	-

[16]https://docs.cohere.com/docs/rerank-2
[17]https://github.com/castorini/pygaggle
[18]https://github.com/PrithivirajDamodaran/FlashRank

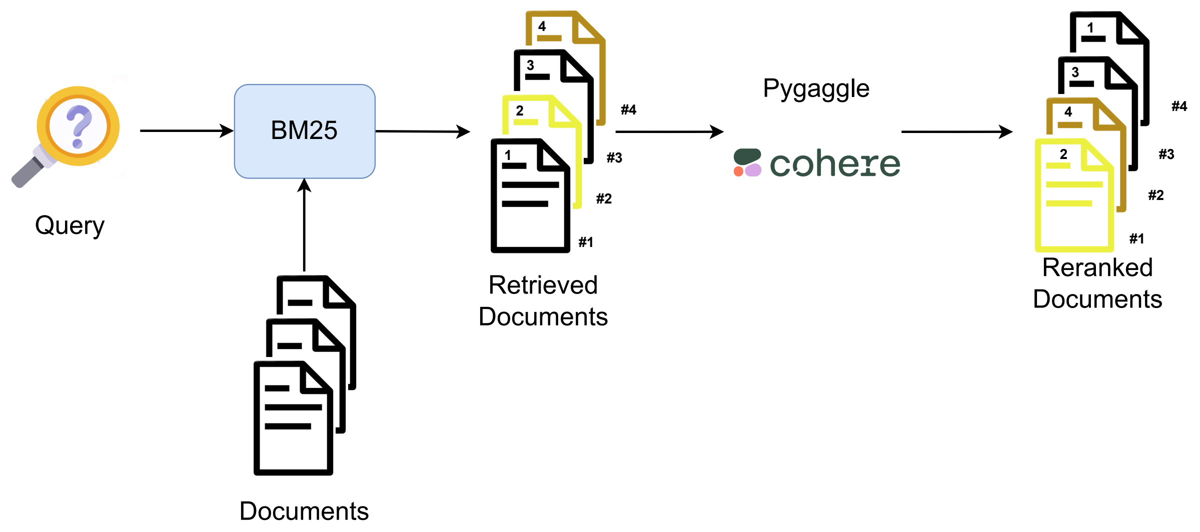


Figure 4: Reranking schema. The relevant documents (yellow and bronze) are sorted towards the first positions of the final retrieved document list considering the scores obtained using Cohere and Pygaggle.

4. Experimental Setup

The experimental setup of the project consists of the following:

- The project source code is available in the Mouse group repository in Bitbucket: https://bitbucket.org/upd-dei-stud-prj/seupd2324-mouse/src/master/.
- The collection used was provided by the CLEF LongEval 2024 Organizers at: https://researchdata.tuwien.at/records/y60e9-k9b51.
- The evaluation tool used is the `trec_eval` script (provided during the course and present in the repository).
- To compute the runs, we used the following hardware:
  - PC 1 - French: Microsoft Windows 11 Home, CPU 13th Gen. Intel i9 (20@5.2GHz), GPU Intel Raptor Lake-P [Iris Xe Graphics].
  - PC 2 - English: Pop!_OS 22.04 LTS, CPU 11th Gen. Intel i7 (8@4.7GHz), GPU Intel TigerLake-LP GT2 [Iris Xe Graphics].

We provided a README file in the group repository with all the instructions for reproducibility.

5. Results and Discussion

In this Section, we present the experimental results obtained and discuss the findings. We structured the presentation of the results by dividing the discussion based on the language of the collection used, i.e., English and French.

5.1. Training Results

We report the results obtained using the implemented system. For clarity reasons, we decided to structure the name of the models used for the runs deciding the such names on the parameter values reported in Table 1, displaying the names in a common structure:

Model: `StemFilter_Tokenizer_Stoplist_QueryExpansionModel_RerankingModel`

English Collection Results. Table 2 displays the MAP and nDCG scores for the runs conducted on the English data collection. As outlined in Section 3, the English dataset was generated through automated translation of the original French queries. Consequently, we noted that the MAP and the nDCG were affected by the quality of these translations. Furthermore, it appears that using query expansion and reranking techniques based on LLMs is not successful enough in resolving the issue of poorly translated queries and documents.

Table 2
MAP and nDCG with the English collection.

Model	MAP	nDCG
Eng-LLama3-70b_Cohere-100-w06	0.2226	0.4017
Eng-Mixtral-8x7b_Pygaggle-Luyu-20-w06	0.2153	0.3968
Eng-LLama3-70b_Pygaggle-Luyu-20-w06	0.2144	0.3955
Eng-LLama3-70b	0.1946	0.3799
Eng-Mixtral-8x7b	0.1933	0.3787

Figure 5 illustrates the interpolated Precision-Recall curve, which can be advantageous in understanding the inverse relationship between the two measures. Hence, the graph highlights the importance of balancing the fraction of retrieved documents that are relevant to the user, the Precision of the system, and the effectiveness of the retrieval system in obtaining all pertinent documents, i.e., the Recall. The curve indicates that the model employing the LLama3-70b and Cohere models for query expansion and reranking, respectively, generally provides a superior Precision Recall trade-off if compared to other models. Conversely, the absence of reranking and stopwords removal appears to be disadvantageous for the system, cf. models Porter_Standard_stopwords-en.txt_LLama3-70b and Porter_Standard_Mixtral-8x7b.

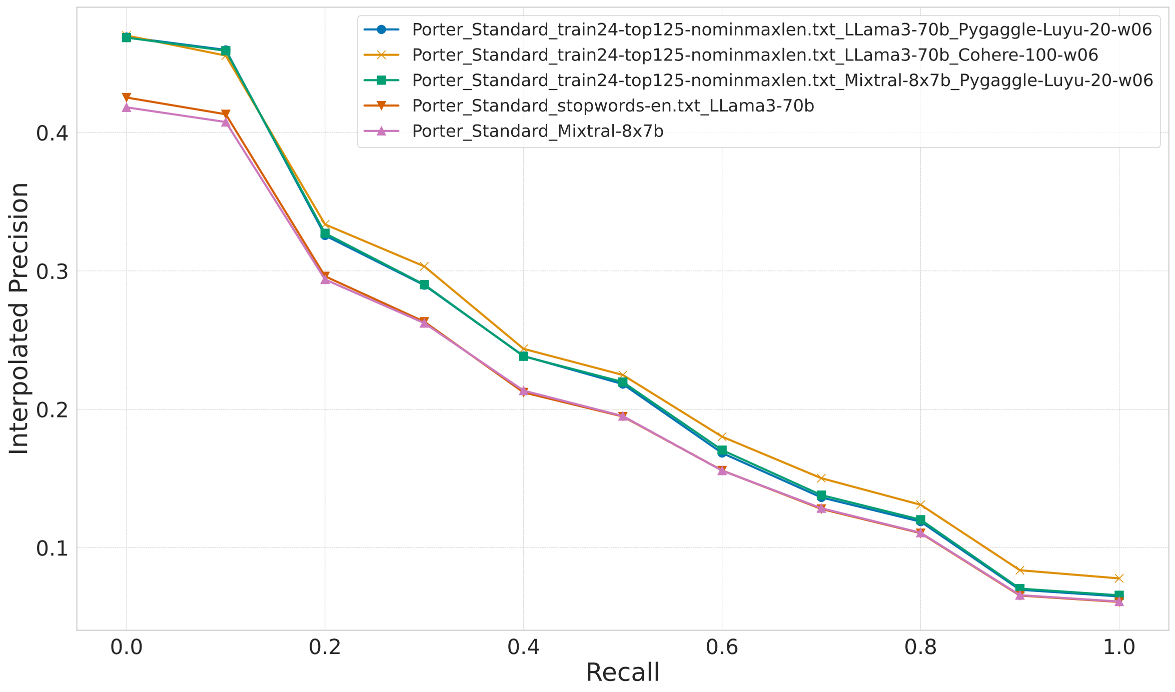


Figure 5: Interpolated Precision Recall graph, English collection. The dataset used is LongEval 2024 Train Collection (cf. 4).

French Collection Results. Using the original French queries positively impacted the system's performance in terms of both MAP and nDCG. Table 3 describes the systems' performance for the different runs using the French collection. The French runs yielded a considerable improvement in the models, with each of them increasing the MAP and nDCG by at least 7% and 9%, respectively. Once again the worst results were presented by the model with no stopwords filtration and reranking, decreasing the best model performance of 5% in terms of MAP and 4% in terms of nDCG. However, such a model reaches better performance if compared to the best model of Table 2, which used the English collection data.

Table 3
MAP and nDCG with the French collection.

Model	MAP	nDCG
French-Light_Standard_train24-top125-nominmaxlen.txt_LLama3-70b_Pygaggle-Luyu-20-w06	0.3127	0.5077
French-Light_Standard_train24-top125-nominmaxlen.txt_LLama3-70b_Cohere-100-w06	0.3107	0.5052
French-Light_Standard_train24-top125-nominmaxlen.txt_Mixtral-8x7b_Pygaggle-Luyu-20-w06	0.3101	0.5055
French-Light_Standard_stopwords-fr.txt_LLama3-70b	0.2921	0.4917
French-Light_Standard_Mixtral-8x7b	0.2622	0.4663

Finally, we also present the Interpolated Precision-Recall curve derived from the French dataset. Once again, we note a consistent pattern of the Precision-Recall trade-off, highlighting the importance of stopwords removal for this collection before starting the search process. It has been observed that the integration of query expansion and reranking, based on LLMs, results in a consistent and outstanding performance across the different models used.

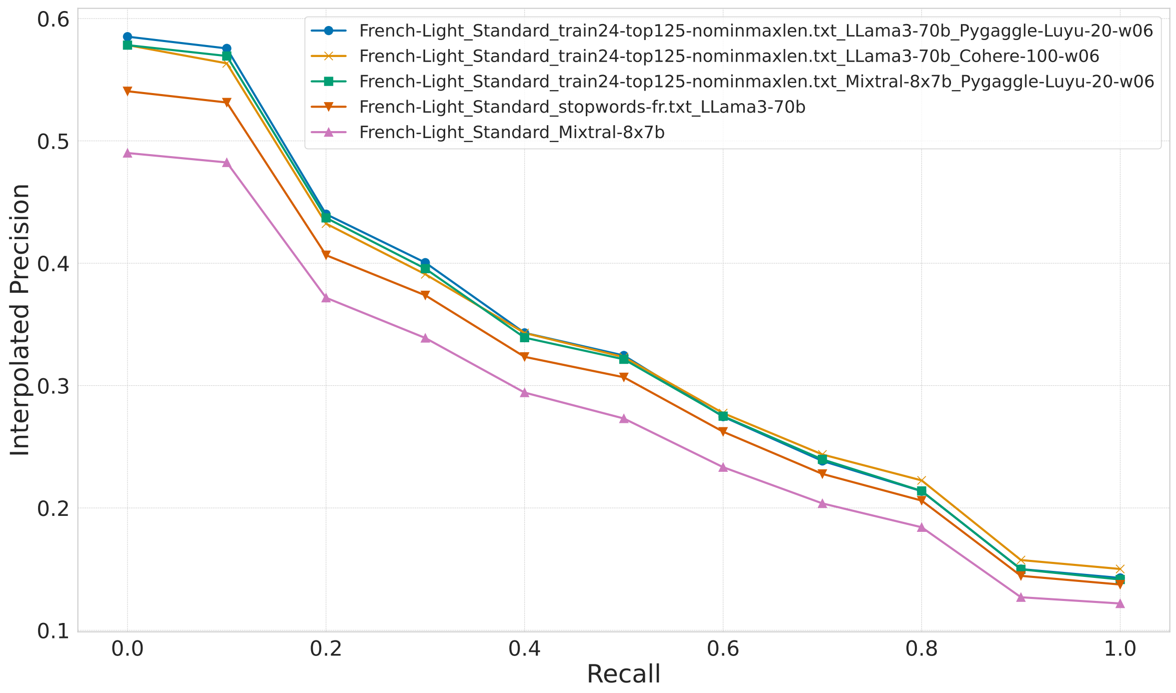


Figure 6: Interpolated Precision Recall graph, French collection. The dataset used is LongEval 2024 Train Collection (cf. 4).

5.1.1. Discussion

Observing the results obtained, we derived that the English collection, composed of automatically translated queries and documents, yielded poor performance outcomes. This lack of effectiveness can be attributed to the automatic translation process, which often fails to translate the query correctly and alters the user's real information needs. In second place, the adoption of a stopwords list had an important impact on the systems' performance: instances where irrelevant words were not removed, did not yield meaningful results, contributing to poor performance even when queries were expanded. As a matter of fact, from the training results, the influence of LLMs on query expansion and reranking is significant in both analyzed scenarios. Across the experiments, the most effective models consistently included query expansion and reranking pipelines based on LLama3, Mixtral, Cohere, and Pygaggle models. These models increased the overall performance in terms of MAP, nDCG, and Precison-Recall trade-off, showcasing the significant potential of leveraging advanced language models for enhancing retrieval relevance.

5.2. Test Results

This Section comprises the statistical tests performed to investigate the performance of the submitted runs on the short and long-term collections. Specifically, we tackle the performance changes and the statistical analysis of the results obtained. Moreover, the ANOVA 2 test is used with a significance level of $\alpha = 0.05$ to assess whether the null hypothesis can be rejected, thereby indicating a significant statistical difference between the results of the given runs. Specifically, the two-way ANOVA is used to determine how two independent variables—namely, the system and a query—impact a dependent variable, which, in our case, is a specific measure. In each ANOVA2 table, we reported **df**,i.e., the degree of freedom in the source, **SS**, i.e., the sum of squares due to the source, **MS**, i.e., the mean sum of squares due to the source, **F**, i.e., the F-statistic and **PR(>F)** that is the p-value. To avoid encumbering, we report only meaningful insights on the measures collected, however, all the analysis and the graphs are available on the study repository. Finally, we performed the pairwise comparison, i.e., the Tukey's *Honestly Significant Difference (HSD)* test, to verify if, among our submitted systems, some significant statistical differences are present. For each multi-comparison graph, we highlighted in blue the best model obtained, in red the models for which there is a significant difference computed by Tukey's HSD test. In gray, models that are not statistically different are reported.

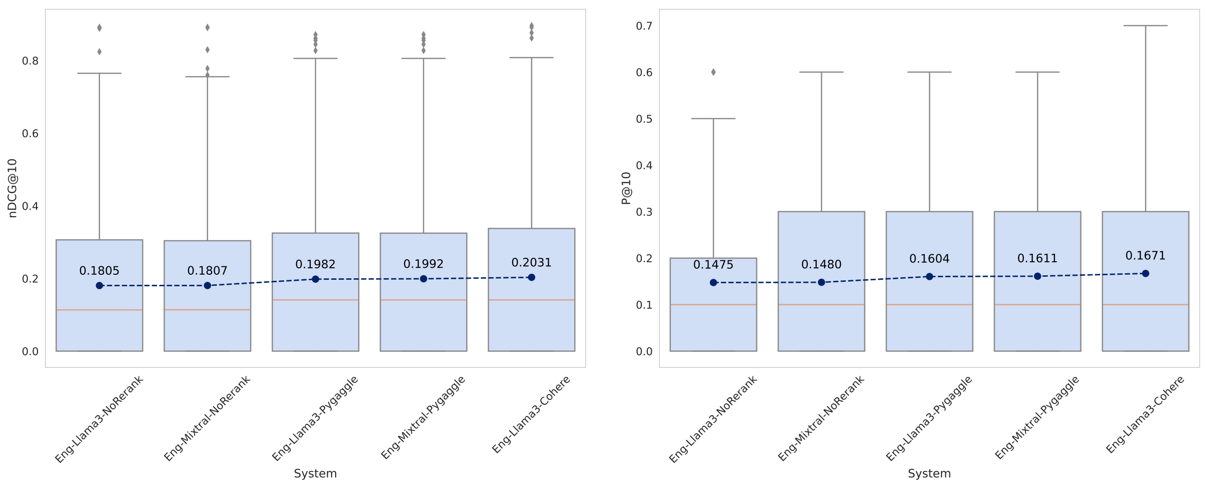
Since our submissions were both using English and French collections, for this final analysis we decided not to remove any of the systems used during the training phase.

5.2.1. English Test Results: Short Term collection

Table 4
Overall results on test set - English Collection - Short term. The Table reports the averages of the performances measured.

	June				
System	*nDCG*	*nDCG@10*	*MAP*	*P@10*	*Recall@1000*
Eng-Llama3-Cohere rerank	0.3060	0.2031	0.1705	0.1671	0.5827
Eng-Llama3-Pygaggle rerank	0.3038	0.1982	0.1666	0.1604	0.5827
Eng-Mixtral-Pygaggle rerank	0.3036	0.1992	0.1664	0.1611	0.5821
Eng-Llama3-NoRerank	0.2914	0.1805	0.1525	0.1475	0.5817
Eng-Mixtral-NoRerank	0.2910	0.1807	0.1527	0.1480	0.5809

Table 4 reports the test results obtained by each of the submitted systems on the June test collection. Upon comparing the results obtained during the training phase, cf. Table 2, it is evident that there is a slight reduction in the effectiveness of the retrieval systems, particularly in terms of MAP and nDCG. Based on the Boxplots obtained from Figure 7, it is evident that the model utilizing Llama3-Cohere query expansion and reranking demonstrates ideal robustness in handling performance losses from training to testing phases, preserving for some topics good results, as indicated by both nDCG@10 and P@10 metrics. A final important remark on the boxplots is that all the systems present a similar trend in terms of median and interquartile range, also showing the presence of outliers for specific queries.



(a) nDCG@10 (b) P@10

Figure 7: Box plot of the nDCG@10 and of P@10 for the English Short term collection.

ANOVA 2 The ANOVA 2 analysis was performed to investigate how different types of systems were able to retrieve the relevant documents for the provided topics and, in addition, to understand how different searched topics, i.e., the queries, performed across these systems. Table 5 shows the ANOVA2 tables of the nDCG@10 and P@10 on the June dataset. The insight that Table 5 is that the *pvalue* of the test is below the threshold $\alpha = 0.05$, implying that there is indeed a significant statistical difference among the systems.

Table 5
Anova 2 Way for nDCG@10 and P@10 considering Topics and Systems for the English Short term collection.

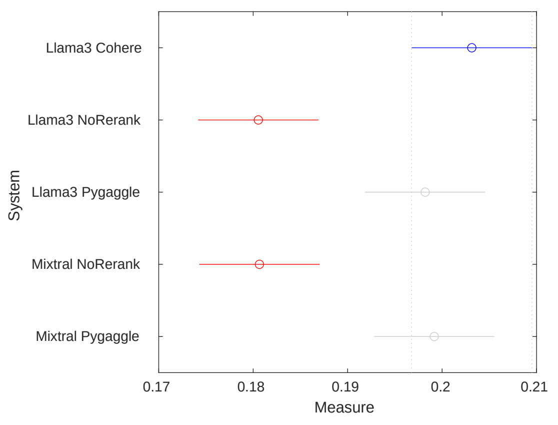
	nDCG@10					P@10				
	df	***SS***	***MS***	***F***	***PR(>F)***	***df***	***SS***	***MS***	***F***	***PR(>F)***
Topics	403	80.5779	0.1999	45.4342	0	403	46.8477	0.1162	53.1366	0
Systems	4	0.1912	0.0478	10.8625	1.0846e-8	4	0.1214	0.0303	13.8747	3.9426e-11
Residuals	1612	7.0940	0.0044	-	-	1612	3.5265	0.0022	-	-
Total	2019	87.8631	-	-	-	2019	50.4956	-	-	-

Tukey's HSD Test The results of Tukey's HSD multiple comparisons are reported in Figure 8. From the pairwise test between the different systems, we can see that for the short-term dataset from the best systems, i.e., the models that use the LLM for query expansion and reranking there are no significant statistical differences, for both nDCG@10 and P@10, while on the other hand, the differences are shown with the models that only uses query expansions models.

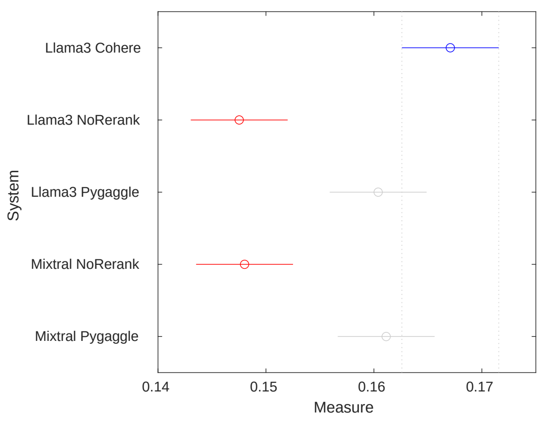
5.2.2. English Test Results: Long Term collection

Table 6 reports the test results obtained by each of the submitted systems on the test collection for the English August collection. Also in this case, the results are sorted based on the nDCG measure, implying that the model that achieved the best nDCG was the Llama3 Cohere model.

From the box plots in Figure 9, once again, we can observe that for both measures, we obtained comparable graphs in terms of median and inner quartiles, showing the presence of outliers topics towards better performance. If we compare the results obtained in Figure 9, with the June test results, Figure 7, what we can gain is the information that the model with the higher mean in terms of nDCG@10 and P@10 is the Llama3 Cohere model, demonstrating the robustness of the system in handling changes through time.



(a) nDCG@10

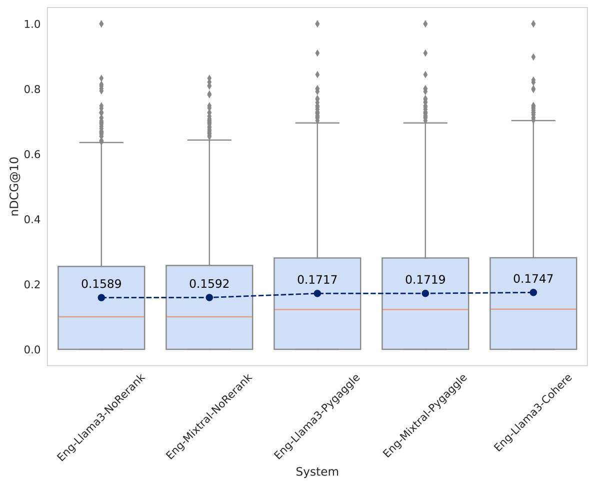


(b) P@10

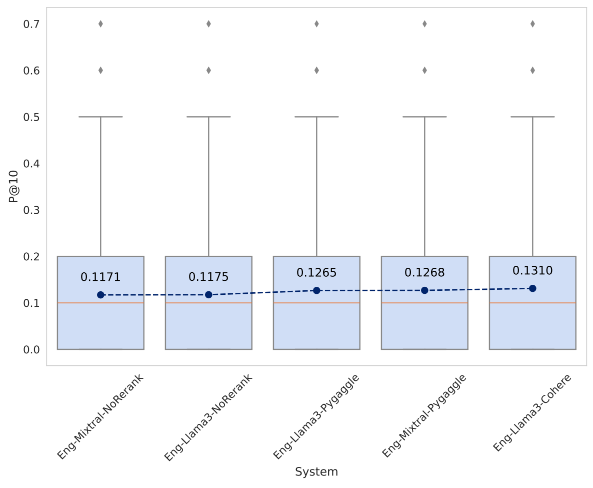
Figure 8: Tukey's HSD test for nDCG@10 and of P@10 in the English Short term collection between the different systems.

Table 6
Overall results on test set - English Collection - Long term. The Table reports the averages of the performances measured.

	August				
System	*nDCG*	*nDCG@10*	*MAP*	*P@10*	*Recall@1000*
Eng-Llama3-Cohere rerank	0.2353	0.1747	0.1265	0.1310	0.4005
Eng-Mixtral-Pygaggle rerank	0.2324	0.1719	0.1238	0.1268	0.4008
Eng-Llama3-Pygaggle rerank	0.2323	0.1717	0.1237	0.1265	0.4005
Eng-Mixtral-NoRerank	0.2254	0.1592	0.1143	0.1171	0.4016
Eng-Llama3-NoRerank	0.2246	0.1589	0.1151	0.1175	0.3996



(a) nDCG@10



(b) P@10

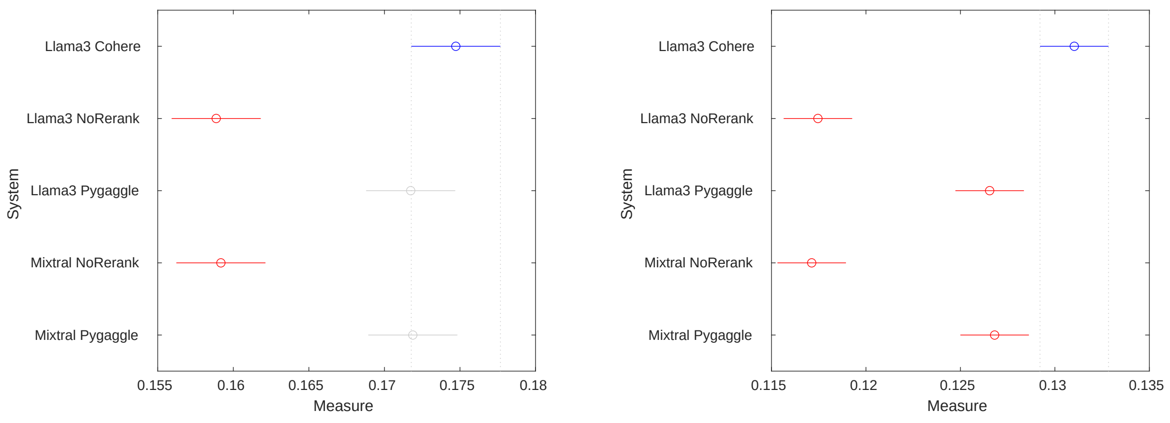
Figure 9: Box plot of the nDCG@10 and of P@10 for the English Long term collection.

ANOVA 2 Even in this case, by using as two independent variables the systems and the topics used for the retrieval process, we wanted to investigate how such variables influenced the performances for the nDCG@10 and the P@10. The ANOVA 2 table, Table 7, provides evidence of a statistically significant difference between the models for all analyzed measures, indicating that the *pvalue* is lower than the threshold α.

Table 7
Anova 2 Way for nDCG@10 and P@10 considering Topics and Systems for the English Long term collection.

	nDCG@10					P@10				
	df	***SS***	***MS***	***F***	***PR(>F)***	***df***	***SS***	***MS***	***F***	***PR(>F)***
Topics	1517	239.9876	0.1582	44.5911	0	1517	114.6827	0.0756	56.3080	0
Systems	4	0.3534	0.0883	24.9026	1.7557e-20	4	0.2331	0.0583	43.4200	5.5860e-36
Residuals	6068	21.5278	0.0035	-	-	6068	0.0013	0.0013	-	-
Total	7589	261.8688	-	-	-	7589	114.9171	-	-	-

Tukey's HSD Test The graphs of Figure 10 represent the pairwise Tukey's HSD test. For the nDCG@10 measure, there is no significant statistical difference among the LLMs reranking-based systems, while on the other hand, the models that do not implement any reranking strategy obtain worse performance and they are found to be statistically different from the best model. However, when it comes to P@10, the statistical difference is found also with the other two query expansion and reranking LLMs based systems: this aspect underlines the importance and the positive impact that query expansion and reranking performed with the latest versions of LLMs architecture have on the search pipeline.



(a) nDCG@10 (b) P@10

Figure 10: Tukey's HSD test for nDCG@10 and of P@10 in the English Long term collection between the different systems.

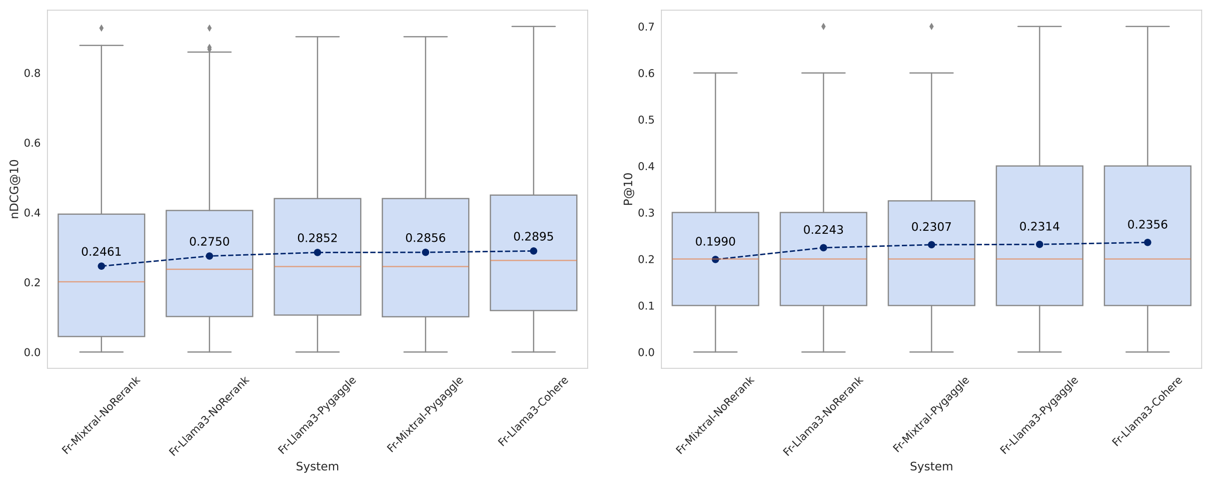
5.2.3. French Test Results: Short Term collection

Table 8
Overall results on test set - French Collection - Short term. The Table reports the averages of the performances measured.

			June		
System	*nDCG*	*nDCG@10*	*MAP*	*P@10*	*Recall@1000*
Fr-Llama3-Cohere rerank	0.3946	0.2895	0.2484	0.2356	0.6615
Fr-Mixtral-Pygaggle rerank	0.3932	0.2856	0.2459	0.2307	0.6615
Fr-Llama3-Pygaggle rerank	0.3923	0.2852	0.2453	0.2314	0.6615
Fr-Llama3-NoRerank	0.3863	0.2750	0.2355	0.2243	0.6639
Fr-Mixtral-NoRerank	0.3672	0.2461	0.2145	0.1990	0.6636

Table 8 reports the test results obtained on the June French test collection by the five submitted systems. Comparing the results achieved in the training phase, cf. Table 3, it is clear that the effectiveness of retrieval systems has slightly decreased. However, considerable values in terms of MAP and nDCG have been achieved, especially compared to those obtained on the English collection. A possible motivation for explaining the fact that the results of the French collection are superior to those obtained from the English collection is because, in the former case, the original documents and queries were used, while in the latter case, an automatic translation has been applied. As a consequence, in the latter case, the results depend heavily on the quality of the translation.

Upon examing the Boxplots depicted in Figure 11, it is possible to understand that the systems with reranking outperformed those without this feature. In particular, the system that employed Llama3 as the query expansion method and Cohere as the rerank model achieved, once again, the most favourable results when compared to the others.



(a) nDCG@10 (b) P@10

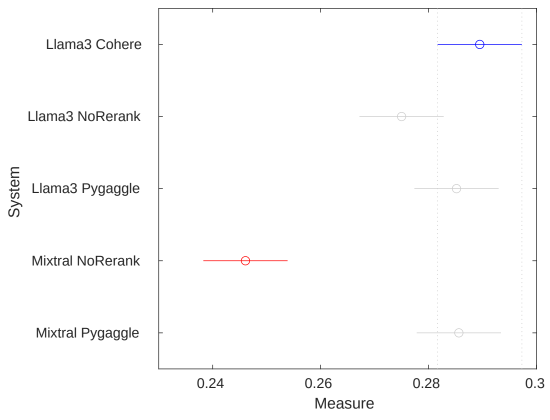
Figure 11: Box plot of the nDCG@10 and of P@10 for the French Short term collection.

ANOVA 2 As previously done for the English June collection, we performed the Anova 2 test to gain a deeper understanding of the topics and systems influence for achieving the computed values. In this case, Table 9, the result was that *pvalue* less than α, thus implying the presence of statistical difference in the analysis of systems and queries.

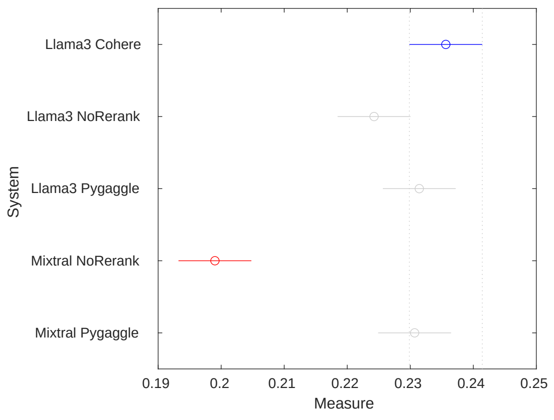
Table 9
Anova 2 Way for nDCG@10 and P@10 considering Topics and Systems for the French Short term collection.

	nDCG@10					P@10				
	df	***SS***	***MS***	***F***	***PR(>F)***	***df***	***SS***	***MS***	***F***	***PR(>F)***
Topics	403	89.7287	0.2226	33.6255	0	403	50.0862	0.1243	34.2320	0
Systems	4	0.5068	0.1267	19.1337	2.1754e-15	4	0.3474	0.0869	23.9247	2.9997e-19
Residuals	1612	10.6739	0.0066	-	-	1612	5.8525	0.0036	-	-
Total	2019	100.9094	-	-	-	2019	56.2861	-	-	-

Tukey's HSD Test The Tukey's HSD test confirmed the hypothesis of the importance of the LLMs query expansion and reranking process: in Figure 12, the best model, represented by the French Llama3 Cohere model, shows a statistical difference in the performance, nDCG@10 and P@10, with the Mixtral NoRerank model. Moreover, the query expansion process performed with the Llama3 model underlines the strength of the model to reach out to the performances of the reranking systems. A possible future direction can be related to an extensive analysis toward more specific research in the adoption of a certain LLM architecture over another for the expansion of a query and the reranking. We leave this aspect as an open issue and possible future work.



(a) nDCG@10



(b) P@10

Figure 12: Tukey's HSD test for nDCG@10 and of P@10 in the French Short term collection between the different systems.

5.2.4. French Test Results: Long Term collection

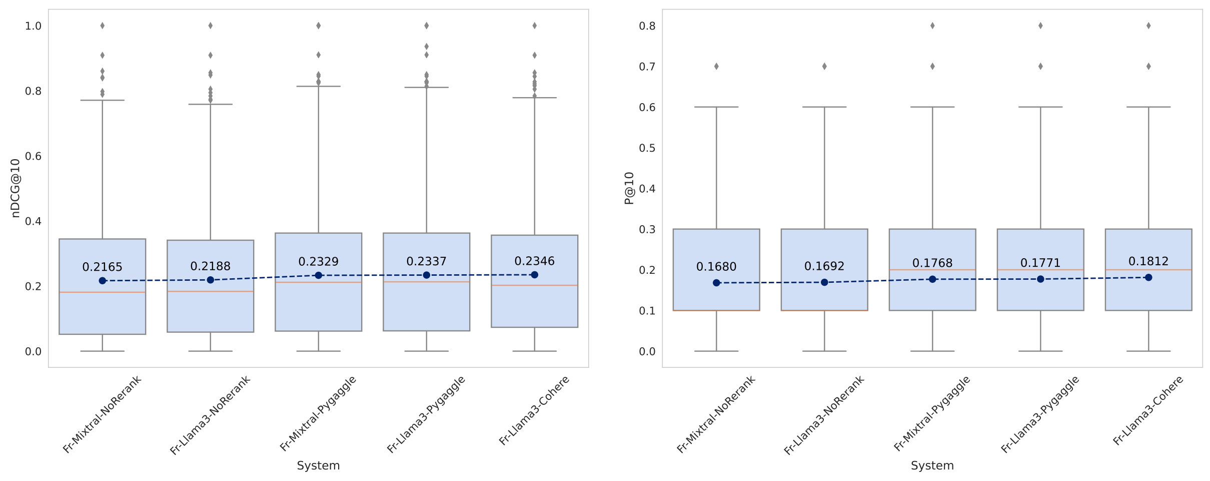
Table 10
Overall results on test set - French Collection - Long term. The Table reports the averages of the performances measured.

	August				
System	*nDCG*	*nDCG@10*	*MAP*	*P@10*	*Recall@1000*
Fr-Llama3-Pygaggle rerank	0.2981	0.2337	0.1752	0.1771	0.4620
Fr-Llama3-Cohere rerank	0.2979	0.2346	0.1741	0.1812	0.4620
Fr-Mixtral-Pygaggle rerank	0.2976	0.2329	0.1748	0.1768	0.4620
Fr-Llama3-NoRerank	0.2877	0.2188	0.1632	0.1692	0.4616
Fr-Mixtral-NoRerank	0.2863	0.2165	0.1615	0.1680	0.4614

Finally, we report the August results for the French collection. Table 10 reports the test results obtained by each of the submitted systems on the test collection. Upon comparing these results with those achieved in the training phase, once again, shown in Table 3, it is evident that the effectiveness of the retrieval systems has experienced a slight decrease. However, notable values in terms of MAP and nDCG have been attained, this time by the Llama3 Pygaggle model, particularly in comparison to those obtained from the English collection. Once again we want to stress the fact that an underlying reason for the better results of the French collections, as opposed to the English ones, may be attributed to the usage of the original documents and queries in the former, while an automatic translation was applied in the latter. Consequently, the quality of the translation significantly influences the results in the latter case.

Moreover, the box plots in Figure 13 show comparable results to the ones provided in the Sections above. However, it is possible to understand that this time, the mean values of the P@10 for the models with query expansion and reranking based on LLMs are below the medians, hence showing a negatively skewed distribution of such results.

ANOVA 2 Table 11 shows the results of the ANOVA 2 for nDCG@10 and P@10. In this case, there are significant differences as the *pvalue* obtained from the two-way test is below the threshold $\alpha = 0.05$.

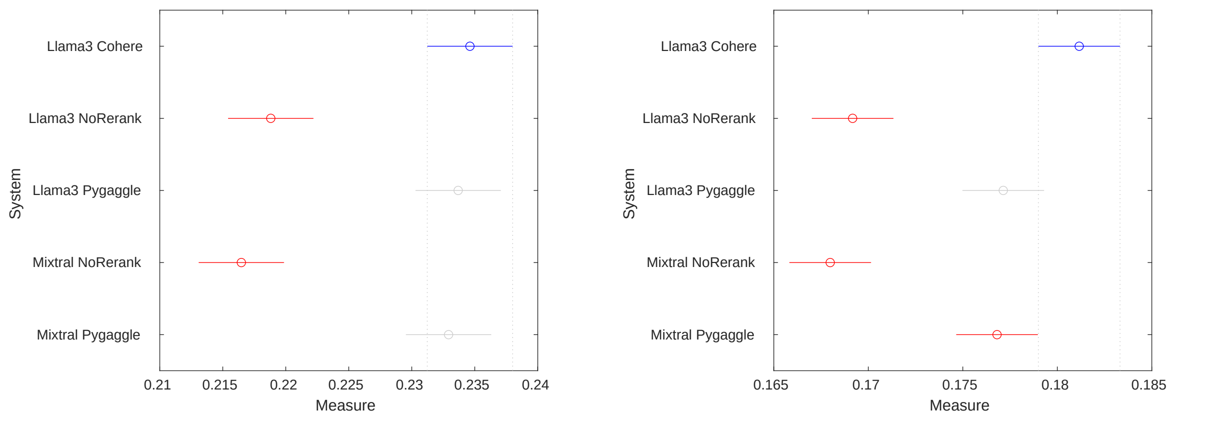


(a) nDCG@10 (b) P@10

Figure 13: Box plot of the nDCG@10 and of P@10 for the French Long term collection.

Table 11
Anova 2 Way for nDCG@10 and P@10 considering Topics and Systems for the French Long term collection.

	nDCG@10					P@10				
	df	***SS***	***MS***	***F***	***PR(>F)***	***df***	***SS***	***MS***	***F***	***PR(>F)***
Topics	1517	267.5830	0.1764	37.6977	0	1517	141.7925	0.0934	49.1038	0
Systems	4	0.4784	0.1196	25.5625	4.1955e-21	4	0.1936	0.0484	25.4237	6.4256e-21
Residuals	6068	28.3925	0.0047	-	-	6068	11.5504	0.0019	-	-
Total	7589		-	-	-	7589	153.5365	-	-	-



(a) nDCG@10 (b) P@10

Figure 14: Tukey's HSD test for nDCG@10 and of P@10 in the French Long term collection between the different systems.

Tukey's HSD Test To conclude the statistical test analysis, we finally implemented a pairwise Tukey's HSD test to get important statistical differences between the models. However, conversely to the case analyzed for the June French collection, cf. Figure 12, the no reranking models present a significant difference with the reranking models in term of nDCG@10, while the test on the P@10 measure underlines once again the robustness of the Llama3 query expansion process showing no differences between the reranking with Cohere and Pygaggle.

6. Conclusions and Future Work

Our work proposed a method for Searching relevant documents based on query expansion and reranking techniques that use the application of LLMs. Our system was able to achieve a MAP of 0.3127 and nDCG of 0.5077 on the French data collection using LLama 3-70b and the Pygaggle-Luyu-20-w06 models for query expansion and reranking.

We observed a significant enhancement in our study when employing the French dataset over the English counterpart. It appears that translations may introduce inaccuracies and inconsistencies, impacting the overall data quality. Moreover, the application of query expansion and reranking based on LLMs shows a promising potential for further investigation considering the results obtained. Furthermore, further related works plan to enhance the effectiveness of our system, exploring the utilization of next-generation LLMs for translation, query expansion and reranking, as they will offer superior capabilities compared to current models in task comprehension and text generation capabilities. Finally, experimenting with various combinations of stoplists and stemming methods could yield even more favourable outcomes, optimizing our system's performance.

References

[1] P. Galuscáková, R. Deveaud, G. G. Sáez, P. Mulhem, L. Goeuriot, F. Piroi, M. Popel, Longeval-retrieval: French-english dynamic test collection for continuous web search evaluation, in: H. Chen, W. E. Duh, H. Huang, M. P. Kato, J. Mothe, B. Poblete (Eds.), Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023, ACM, 2023, pp. 3086–3094. URL: https://doi.org/10.1145/3539618.3591921. doi:10.1145/3539618.3591921.

[2] E. M. Voorhees, T. Alam, S. Bedrick, D. Demner-Fushman, W. R. Hersh, K. Lo, K. Roberts, I. Soboroff, L. L. Wang, TREC-COVID: constructing a pandemic information retrieval test collection, SIGIR Forum 54 (2020) 1:1–1:12. URL: https://doi.org/10.1145/3451964.3451965. doi:10.1145/3451964.3451965.

[3] T. Qwant, Qwant search engine, 2013. URL: https://about.qwant.com/en/, accessed on April 17, 2024.

[4] D. Harman, M. Liberman, Tipster complete, 1993. URL: https://catalog.ldc.upenn.edu/LDC93T3A, accessed on May 3, 2024.

[5] G. Antolini, N. Boscolo, M. Cazzaro, M. Martinelli, S. Safavi, F. Shami, N. Ferro, Seupd@clef: Team CLOSE on temporal persistence of IR systems' performance, in: M. Aliannejadi, G. Faggioli, N. Ferro, M. Vlachos (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), Thessaloniki, Greece, September 18th to 21st, 2023, volume 3497 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 2368–2395. URL: https://ceur-ws.org/Vol-3497/paper-192.pdf.

[6] O. AI, Text generation models openai, 2022. URL: https://platform.openai.com/docs/models/overview, accessed on May 3, 2024.

[7] E. Bolzonello, C. Marchiori, D. Moschetta, R. Trevisiol, F. Zanini, N. Ferro, Seupd@clef: Team FADERIC on A query expansion and reranking approach for the longeval task, in: M. Aliannejadi, G. Faggioli, N. Ferro, M. Vlachos (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2023), Thessaloniki, Greece, September 18th to 21st, 2023, volume 3497 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023, pp. 2252–2280. URL: https://ceur-ws.org/Vol-3497/paper-188.pdf.

[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (2020) 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

[9] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: J. Burstein, C. Doran, T. Solorio (Eds.), Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics:

Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, 2019, pp. 4171–4186. URL: https://doi.org/10.18653/v1/n19-1423. doi:10.18653/V1/N19-1423.

[10] J. Hadley, Unicode text segmentation, 2023. URL: https://unicode.org/reports/tr29/, accessed on May 3, 2024.

[11] M. F. Porter, An algorithm for suffix stripping, Program 14 (1980) 130–137.

[12] R. Krovetz, Viewing morphology as an inference process, in: R. R. Korfhage, E. M. Rasmussen, P. Willett (Eds.), Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993, ACM, 1993, pp. 191–202. URL: https://doi.org/10.1145/160688.160718. doi:10.1145/160688.160718.

[13] J. B. Lovins, Development of a Stemming Algorithm, Mechanical Translation and Computational Linguistics 11 (1968) 22–31.

[14] J. Savoy, A stemming procedure and stopword list for general french corpora, Journal of the American Society for Information Science 50(10), 944-952. (2009).

[15] D. Gene, A. Suriyawongkul, M. Pukhalskyi, B. Solomon, Stopwords iso, 2020. URL: https://github.com/stopwords-iso/stopwords-iso, accessed on April 18, 2024.

[16] K. Spärck Jones, S. Walker, S. E. Robertson, A probabilistic model of information retrieval: development and comparative experiments – Part 1, Information Processing & Management 36 (2000) 779–808.

[17] K. Spärck Jones, S. Walker, S. E. Robertson, A probabilistic model of information retrieval: development and comparative experiments – Part 2, Information Processing & Management 36 (2000) 809–840.

[18] K. Spärck Jones, A statistical interpretation of term specificity and its application in retrieval, Journal of Documentation 28 (1972) 11–21.

[19] F. J. Damerau, A technique for computer detection and correction of spelling errors, Commun. ACM 7 (1964) 171–176. URL: https://doi.org/10.1145/363958.363994. doi:10.1145/363958.363994.

[20] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, G. Lample, Llama: Open and efficient foundation language models, CoRR abs/2302.13971 (2023). URL: https://doi.org/10.48550/arXiv.2302.13971. doi:10.48550/ARXIV.2302.13971. arXiv:2302.13971.

[21] A. Q. Jiang, A. Sablayrolles, A. Roux, A. Mensch, B. Savary, C. Bamford, D. S. Chaplot, D. de Las Casas, E. B. Hanna, F. Bressand, G. Lengyel, G. Bour, G. Lample, L. R. Lavaud, L. Saulnier, M. Lachaux, P. Stock, S. Subramanian, S. Yang, S. Antoniak, T. L. Scao, T. Gervet, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mixtral of experts, CoRR abs/2401.04088 (2024). URL: https://doi.org/10.48550/arXiv.2401.04088. doi:10.48550/ARXIV.2401.04088. arXiv:2401.04088.

[22] T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, et al., Gemma: Open models based on gemini research and technology, CoRR abs/2403.08295 (2024). URL: https://doi.org/10.48550/arXiv.2403.08295. doi:10.48550/ARXIV.2403.08295. arXiv:2403.08295.