SEUPD@CLEF: Team DAM on Reranking Using Sentence Embedders

Notebook for the LongEval Lab at CLEF 2024

Alberto Basaglia[1], Andrea Stocco[1], Milica Popović[1] and Nicola Ferro[1]

[1]University of Padua, Italy

Abstract

This report gives an overview of the system developed by Team DAM for Task 1 of the LongEval Lab at CLEF 2024. The team members are students enrolled in the Computer Engineering master's program at the University of Padua. The team developed an information retrieval system which is then used to perform queries on a corpus of documents in French language, or in their translated English version.

Keywords

CLEF, LongEval 2024, Information Retrieval, Search Engine, Documents Retrieval, Temporal Persistence, Reranking, Word Embeddings

1. Introduction

Nowadays, online searching for all types of information has become part of people's daily routines. Billions of users worldwide expect to find needed information quickly and accurately. A search engine (SE) is a software that helps people satisfy such a need using queries to express an information need. Since the number of web pages has been rapidly increasing, there are considerable challenges that this type of software faces. One of the main challenges is the variability of performance of the system over time. That is why LongEval Lab, organized by the Conference and Labs of the Evaluation Forum (CLEF), aims to solve this problem by encouraging participants to develop information retrieval (IR) systems that can adapt to the evolution of corpus over time.

The paper is organized as follows: 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 Work

Sentence Embedders have been extensively used for the reranking phase of information retrieval systems for many years [1] [2]. Recent research has continued to demonstrate the effectiveness of reranking approaches. For instance, Bolzonello et al. (2023) utilized a reranking-based approach successfully, further validating its efficacy in enhancing retrieval performance [3].

3. Methodology

Our approach is based on the usage of finely tuned off-the-shelf components provided by Apache Lucene. In addition to those, a reranking phase based on Sentence Embedder has been implemented. Our idea was to use a model fine-tuned mostly on online data (Reddit comments, citation pairs and WikiAnswer, just to name a few) to try to encode the meaning of topics and documents in an effective

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

alberto.basaglia@studenti.unipd.it (A. Basaglia); andrea.stocco.8@studenti.unipd.it (A. Stocco); milica.popovic@studenti.unipd.it (M. Popović); nicola.ferro@unipd.it (N. Ferro)

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

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

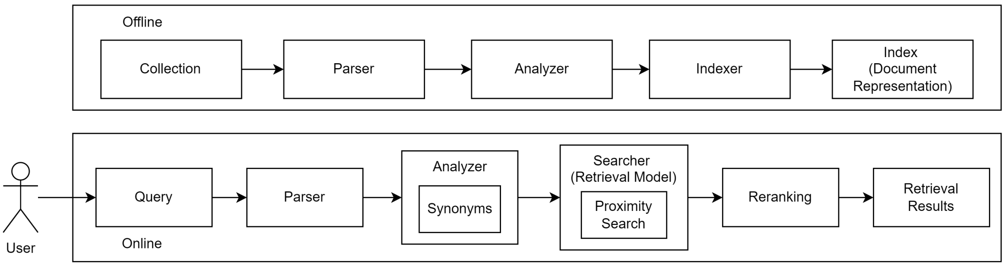


Figure 1: SE architecture

way. Furthermore, we tried to improve one of the works from the previous year which used reranking to improve the IR system performance (Enrico Bolzonello et al. [3]). We used a similar approach based on reranking a small chunk of documents, but using different sentence embedders and a different analyzer pipeline.

In this section we will cover the methodology that has been used to develop our IR system. In order to better understand the complete IR system we developed, all the system's components will be explained using the diagram in Figure 1. The diagram shows the main components of a SE as well as the differentiation between offline and online components.

3.1. Apache Lucene

To develop our IR system, we used Apache Lucene version 9.10.0 [1].

The Apache Lucene project develops open-source search software. It is a high-performance, full-featured SE library written entirely in Java. This library provides a robust and scalable set of tools to developers building efficient IR systems [4]. Thanks to Apache Lucene, we've been able to handle vast amounts of documents with ease, using powerful, accurate, and efficient search algorithms. Moreover, its active community and frequent updates ensure that developers have access to the latest advancements and optimizations in the field of IR.

3.2. Parsing

First of all, to create a fast and reliable IR system, it is necessary to parse the data we want to run our queries on. For this task, LongEval releases the corpus of documents in two different formats, TREC and JSON. We decided to use JSON files.

The corpus came divided into more files, each of them containing a JSON array of documents. The structure of a document is shown in Figure 2.

Figure 2: Structure of a document

```
{
    "docno": "...",
    "text: "...."
}
```

It was thus necessary to write a parser able to read the documents efficiently from the disk.
In order to create an efficient parser for reading documents from the LongEval corpus in JSON format, several key components were implemented:

[1] Downloaded from: https://lucene.apache.org/core/downloads.html

- **File Parser**: a file parser is responsible for reading JSON files containing arrays of documents efficiently. This component iterates through each line of the file and extracts the JSON objects which represent a single document;
- **Document Model**: a document model defines the structure of a document in the corpus. In this case, each document consists of two fields: *docno* (document number) and *text* (document text), as shown in Figure 2. This model is used to deserialize JSON objects into Java objects during parsing;
- **JSON Deserialization**: the parser implements a JSON deserialization library, Jackson [5], to convert JSON objects into Java objects, this way it is easier to manipulate and access the document data;
- **Iterator Implementation**: the parser also implements the Iterator interface to go through the documents in the corpus. This allows efficient and sequential processing of one document at a time, without loading the entire corpus into memory.

3.3. Analyzing

Once the parsing of the documents has been set up, the very next step is analyzing the documents, in general, consisting of:

- **Tokenization**: we split the documents into tokens, that are going to be our "unit" of computation in the system;
- **Stopword removal**: a predefined list of words, considered to be useless in the context of search are removed. An example of these words is articles: they appear in every document so they do not help to discriminate documents;
- **Stemming**: we reduce words to their root or base form to improve search results by capturing variations of the same word.

These are the techniques that have been used in at least one of our experiments:

- **StandardTokenizer**
- **StopFilter**
- **ICUFoldingFilter**
- **LengthFilter**
- **SnowballFilter**

While the techniques mentioned above are valid for both English and French analyzing, some techniques specific to a language have been implemented.

English

- **EnglishPossessiveFilter**
- **KStemFilter**

French

- **FrenchLightStemmer**
- **ElisionFilter**

3.4. Indexing

Once the documents analysis is completed, the next step is indexing. It is a crucial phase, since it creates a searchable database, the index, that contains essential metadata about parsed documents. This metadata includes details like the words and phrases within each document, their frequency, and their location within the document. By structuring documents in this manner, we improve retrieval efficiency, enabling users to search for documents based on keywords or phrases with ease. In order to do that, we indexed the parsed documents through an inverted index, using two different *fields*:

- **DocNo**
- **Content**

So basically, every parsed document is saved in the indexer with these two fields, one used to identify the document itself and the other representing its whole content.

3.5. Searching

The Searcher serves as the component responsible for interpreting input queries and searching through indexed documents to identify those that fit best (*"best match"*) with the query. Then it retrieves these documents and presents them back to the user.

3.5.1. BM25

The next step is to fetch the pertinent documents based on the given queries: this involves identifying the most similar documents to our queries using various scoring functions. So we assign scores to each document in our collection, ranking them from highest to lowest. The highest-ranked document is presumed to be the most relevant to the given query.
The BM25 ranking function, that belongs to the “BM family” of retrieval models (*BM* stands for *Best Match*), in addition of being simple and effective, seems to be very competitive compared to more modern techniques [6]. In Section 5 we used $k_1 = 1.2$ and $b = 0.75$ as parameters for BM25 (the default ones from Apache Lucene).

3.5.2. Queries

Queries are the bridge between user information needs and the underlying document corpus. LongEval provides query datasets in TSV (Tab-Separated Values) format, structured to include query identifiers (*num*) and corresponding textual queries *text*. Each line in the TSV file represents a single query, with the query identifier and text separated by a tab character. Follows an example from LongEval 2024 Test Collection [7]:

```
q062228 aeroport bordeaux.
```

Once TSV queries are loaded and parsed they’re transformed into an Object, that stores *num* and *text*. This is then submitted to the index searcher for retrieval of relevant documents, generating ranked lists of documents based on their relevance to each query. Before submitting them to the actual searcher, queries are parsed using Lucene Query Parser [8], since this package also provides many powerfull tools to modify query terms and implement stategies like fuzzy search, proximity search and term boosting.

3.5.3. Proximity Search

In order to improve the performance of our IR system, one possibility can be using *proximity search*, which allows us to search for a document based on how closely two or more search terms of the query appear in the document. The distance between the two terms is given by a parameter *k*, which depends on the context and the length of the documents. For example, the query "red brick house" could be used to retrieve documents that contain phrases like "red house of brick" or "house made of red brick", while in the meantime avoiding documents where the words are scattered or spread across. Later, we will discuss the improvements given by this kind of search.

3.5.4. Synonyms

Another way to improve the performance of our IR system is using *query expansion*. Query expansion is a technique that consists in reformulating the queries to better match relevant documents. There are several ways to perform it; the one we tried was *synonym query expansion*. With this approach, each query term is expanded with its own synonyms. To find all the synonyms of every english word we used *Wordnet*, that is a large lexical database of English containing all the words (nouns, verbs, adjectives, ...) grouped into sets of cognitive synonyms [10]. The same methodology was also applied to all other Europian languages in *EuroWordNet* project. A list of words with associated synonyms, retrieved from these databases, is available in our repository for both English and French. This approach does not always lead to improvements, indeed there are cases that lead to worse system performance. The results of our experiment will be shown in section 5.

3.5.5. Reranking

After the searching process has retrieved the highest ranked documents with respect to the employed criteria, we can apply a second phase of ranking. This phase is not going to look through all the documents again, but instead it is going to work with the documents the first phase retrieved. In our case, we will take only the first k documents for efficiency reasons. The value of k will be discussed later. The following diagram shows the process flow happening during the reranking phase.

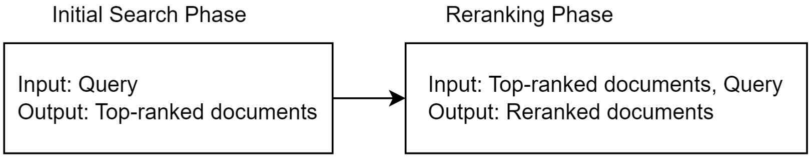

Initial Search Phase
Input: Query
Output: Top-ranked documents
Reranking Phase
Input: Top-ranked documents, Query
Output: Reranked documents

Figure 3: Reranking process flow

The *reranking* approach we use is based on machine learning, specifically on *sentence embedding models.*

Essentially, we employ a pre-trained model that maps text to a vector in a multidimensional space. For each one of the topics we are processing, we compute its vector. Then we take the first k documents retrieved by Lucene's searcher and compute their vector. We then compute a score for each match using the dot product. The resulting value is used to rerank the documents retrieved by the search by increasing their score accordingly.

For each one of the k documents, its updated score will be computed as follows:

$$score_i \leftarrow score_i + R \cdot \mathrm{sim}\left(\mathrm{emb}(q), \mathrm{emb}(d_i)\right).$$

Where R is a coefficient that is used to decide how much to value the output of the sentence embedder; *sim* is the function that computes the similarity between the two vectors. In our system we use one provided by the `sentence_transformers` python package [11]; *emb* computes the embedding of a piece of text. In our example, it is used to compute the vector for the query and for the i-th document. This process will be applied to all the queries.

The model we used is `all-mpnet-base-v2` and it is freely available from HuggingFace [12]. It is trained mostly on English data, from *A Repository of Conversational Datasets* by Henderson et al. [13], so we expect a greater increase in performance when using it for the translated documents. As we will see, this is the case.

The architecture of the model is based on Microsoft's MPNet, which stands for Masked and Permuted Language Modeling [14]. MPNet combines the best parts of BERT and XLNet. It uses a special training [11] that mixes up the order of words to learn their connections while still understanding the context from both directions, like BERT does. This hybrid approach allows MPNet to learn contextual representations more effectively, with improved performance on various natural language understanding tasks as the result. The model is formed by 12 transformer layers, with each one of them having 768 hidden units and 12 attention heads. Given its training on extensive English datasets from Henderson et al.'s repository, `all-mpnet-base-v2` excels in tasks involving English text, and as demonstrated, it shows significant improvements when applied to translated documents. As we will se in Section 5, it will allow us to improve our information retrieval system on the original French documents as well.

As a similarity function we used the `dot_score` function from the `sentence_transformers` package. This function computes the dot product between the two vector associated with the documents [11].

Figure 4 illustrates the reranking algorithm that has been implemented, detailing the number of documents after each phase. To clarify, *N* represents the number of top-ranked documents retrieved in the initial search phase, while *k* has already been mentioned. As it can be seen from the diagram, only scores for the first *k* documents are updated, while the remaining *N-k* documents aren't reranked.

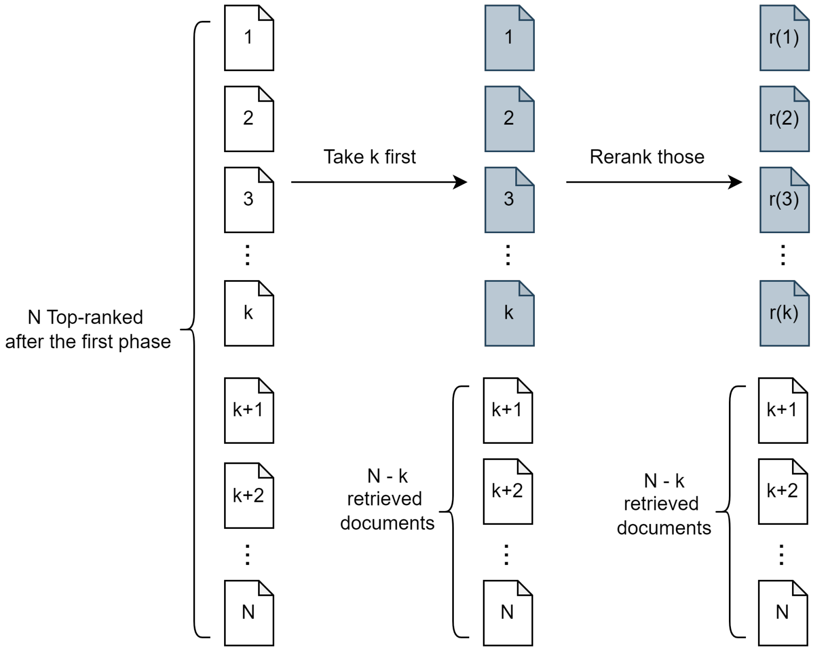


Figure 4: Reranking procedure with the number of documents after each phase

As we previously said, to interact with the model we used the `sentence-transformers` library in python, and in order to interact with python from the java code, we use a *flask* HTTP server [15]. We will then use HTTP requests from the Java code to get the score of a document with respect to a query.

Althought this adds a little overhead, it was much simpler than interacting with the sentence embedder directly from Java.

All the experiments concerning the reranker are available in Section 5.4.

3.5.6. N-gram word model

This section briefly explains another technique for improving SE results, called the N-gram word model. The idea behind it is that phrases are particularly significant in the IR field. Specifically, statistics show that most two or three-word queries are phrases. Because of this, it is important to consider multiple words as phrases rather than independent words. However, the impact of using phrases can be complex, so it is essential to be careful when using them. Let's consider the following definition of the word *phrase*: Phrase is any sequence of n words, also known as an n-gram. Sequences of two words are called *bigrams*, while sequences of three words are called *trigrams*. The greater the frequency of occurrence of a word n-gram, the higher the probability that it corresponds to a meaningful phrase in the language. In the context of IR, n-grams are used to index and retrieve documents based on user queries [16].

4. Experimental Setup

The setup used to run the experiments is the following:

- The system has been run on the collections available from clef-longeval.github.io/data/. All the evaluation has been performed on the 2024 Training Set;
- To evaluate the performance of the SE we used trec_eval 9.0.7, available at trec.nist.gov/trec_eval/;
- The repository with the code can be found at bitbucket.org/upd-dei-stud-prj/seupd2324-dam/.

We ran most of the experiments on the following hardware:

- Intel i7 6700k

- Msi GTX 970
- 16GB DDR4 memory

For reproducibility reasons, the runs were also submitted to TIRA [17]. Since some of the experiments utilize the reranker, the container is given access to a GTX 1080 video card.

4.1. TIRA

TIRA is a key platform for research in information retrieval, designed to facilitate blinded and reproducible experiments. Generally, studies in this field suffer from a lack of reproducibility, since typically only test collections and research papers are shared, requiring third parties to rebuild software to test new datasets. To address this, TIRA has been upgraded to ease task setup and software submission, scaling efficiently from local setups to cloud-based systems using parallel CPU and GPU processing. Overall, TIRA enhances the conduct of AI experiments, ensuring both secrecy and repeatability, and improving the reliability and progression of research in information retrieval.
In particular for Information Retrieval TIREx (Information Retrieval Experiment Platform) [18] has been developed. It integrates ir_datasets, ir_measures, and PyTerrier [19] with TIRA to promote more standardized, reproducible, scalable, and even blinded retrieval experiments.

5. Results and Discussion

Before exploring the results our system is able to obtain, it is useful to list all the possible components we can make use of and assign them a keyword. Such keyword will be used in the name of the run to identify it.

- **FR**: if the indexing and searching process was done on the French version of the documents.
- **EN**: if the indexing and searching process was done on the English version of the documents.
- **Snowball**: if the Snowball stemmer was used.
- **Krovetz**: if the Krovetz stemmer was used.
- **FrenchLight**: if the FrenchLight stemmer was used.
- **Poss**: if the English possessive filter was used.
- **Elision**: if the ElisionFilter was used.
- **Stop**: if the list of stopwords was used. For the English documents we used a default one provided by Lucene. For the French documents we used a custom list that is available in our repository. It is important to note that for french documents we used both lists. This was done because the documents contain some paragraphs in English language.
- **ICU**: if ICU folding was used.
- **Prox**: if proximity search was used. If this keyword is present, we will write also the distance parameters, as explained in Section 3.5.3. For example, if proximity search at a distance of 50 characters is employed, we will write `Prox(50)`.
- **Reranking**: if reranking was used. This will come with a parameter too: the k value we discussed in Section 3.5.5. The R parameter instead will take a fixed value of 5. As an example, if the system reranks the first 50 documents, we will write `Reranking(50)`.
- **Syns**: if the query expansion technique using synonyms was used.
- **Shingles**: if word N-grams are being used. In our case we will generate ngrams of length 2 and 3.

All runs also employ a filter that discards all tokens shorter than 2 characters and longer than 20. Other than that, by default, all the tokens are transformed into lowercase. All the experiments have been run using BM25 with default parameters as the ranking function.

The two main metrics we are going to use to compare runs are the Normalized Discounted Cumulative Gain (nDCG) and the Mean Average Precision (MAP). The reason why we chose these metrics is that they are widely used in IR tasks, since they offer a comprehensive understanding of the effectiveness of retrieval systems. Additionally, they are simple and easy to understand and interpret. nDCG helps measure how close the system's output is when compared to an ideal run, where all the items are sorted in decreasing order of relevance. Moreover, since nDCG is a normalized metric, it enables fair comparisons between different lists of varying lengths and relevance distributions [20]. MAP, on the other hand, calculates the average precision (AP) across all relevant documents, giving insight into the system's overall precision and recall. MAP decreases more rapidly if there are non-relevant items at the top [21]. By deciding to calculate both previously mentioned metrics, we can assess the system's performance from different perspectives.

5.1. Baseline Lucene performance

To set a baseline for our next runs, we ran a baseline Lucene configuration on both the French and the English documents. This baseline configuration will just use the standard tokenizer, the Snowball stemmer, the lower case filter and the BM25 as retrieval model.

The results we obtained are shown in Table 1.

Table 1
Baseline performance

Language	nDCG	MAP
French	**0.4563**	**0.2553**
English	0.3681	0.1842

As we can see, the performance on the "original" French dataset is better than the on on the translated English dataset. *nDCG* and *MAP* are 23.96% and 38.59% better respectively.

This could be explained by the fact that the automated translation is not very accurate and some information is lost in the process.

5.2. Choice of stemmer

As described in Section 3, there are several stemmers that can be used to derive word roots. We made some runs to decide the best one for English and French respectively. For both collections, the corresponding stoplists were used to conduct the experiments.

Table 2
Effect of stemmers on French collection

Stemmer	nDCG	MAP
SnowBall	0.4566	0.2573
FrenchLight	**0.4633**	**0.2626**

FrenchLight stemmer, compared with SnowBall, improves the performance of our system, incrementing both nDCG and MAP of 1.47% and 2.06%, respectively.

Table 3
Effect of stemmers on English collection

Stemmer	nDCG	MAP
SnowBall	**0.3729**	**0.1881**
Krovetz	0.3661	0.1823

In this case, SnowBall stemmer improves the performance of our system compared to Krovetz. Both nDCG and MAP increase of 1.86% and 3.18%, respectively.

5.3. Synonyms

In this section we are going to talk about synonym query expansion. We tried this approach both for English and French. The experiment setup was:

- english stoplist, possessive filter and SnowBall stemmer for English collection.
- french stoplist, elision filter, ICU filter and FrenchLight stemmer for French collection.

The synonyms included in the queries are analyzed with the same analyzer used for indexing the documents and, furthermore, are assigned a weight of 0.7 to reduce their significance compared to the words originally present in the queries. The results are reported in Tables 4 and 5.

Table 4
Effect of synonym query expansion on English collection

Synonyms	nDCG	MAP
Yes	0.3605	0.1798
No	**0.3728**	**0.1880**

Table 5
Effect of synonym query expansion on French collection

Synonyms	nDCG	MAP
Yes	0.4648	0.2581
No	**0.4776**	**0.2733**

As we can see, in both cases the synonym query expansion worsened the performance of our information retrieval system. One of the reasons why this is happening could be that synonyms that are not related to the queries are still included in them because, maybe, they have the same root. So, in the following experiments we completely discard synonyms to focus more on other methods to improve the effectiveness of our system.

5.4. Choice of the reranker parameter

Since, as it was covered in Section 3.5.5, we need to decide a threshold for the number of documents to undergo reranking, we made some runs to decide the best value.

We will use the reranker for both experiments with the English documents and French documents, so we tried it on both the collections. In addition to the techniques used in the previous experiment, we also added a stopword list.

For the French collection the stopword list, ICU folding and the elision filter have been used as well. The runs on the English collection made use of the English possessive filter.

Table 6
Effect of the reranker on the French collection

k	nDCG	MAP
0	0.4776	0.2733
25	0.4832	0.2883
50	0.4877	0.2942
100	0.4904	0.2976
150	0.4917	**0.2986**
200	**0.4918**	0.2985

As we can see, the reranker allow us to improve the performance of the retrieval system. Between the run with no reranking ($k = 0$) and the one with $k = 200$, we notice an increase in the nDCG of 2.97%.

The best value of MAP occurs instead for $k = 150$, with an increase of 9.25%.

Table 7
Effect of the reranker on the English collection

k	nDCG	MAP
0	0.3728	0.1880
25	0.3898	0.2094
50	0.3947	0.2167
100	0.3970	0.2195
150	0.3982	0.2206
200	**0.3990**	**0.2212**

For what concerns the English collection, the best result is obtained with $k = 200$, with an increase of nDCG and MAP of, respectively, the 7.03% and 17.66%.

5.5. Training results

In this section we are going to compare the best configurations for various parameters in both French and English languages. These configurations represent the runs we have chosen to submit to the LongEval Conference:

- The best system that doesn't use reranking for English language;
- The best system for English language, using reranking and SnowBall stemmer;
- The best system that doesn't use reranking;
- The best overall system. This is the system that achieved the best results using reranking and other parameters discussed in the previous sections;
- A system using word N-grams.

The results of these systems are displayed in the following Table 8.

Table 8
Performance of the submitted configurations on the training dataset

Label	System	nDCG	MAP
System 1	EN-Stop-SnowBall-Poss-Prox(50)	0.3820	0.1969
System 2	EN-Stop-SnowBall-Poss-Prox(50)-Reranking(200)	0.3963	0.2147
System 3	FR-Stop-FrenchLight-Elision-ICU-Prox(50)	0.4914	0.2925
System 4	FR-Stop-FrenchLight-Elision-ICU-Prox(50)-Reranking(150)	**0.5024**	**0.3079**
System 5	FR-Stop-FrenchLight-Elision-ICU-Shingles-Prox(50)-Reranking(150)	0.4774	0.2758

One first notable observation is that our search engine generally performs better on French documents. As we previously noted, this trend may be attributed to the fact that the original corpus was written in French and later translated into English.
A further analysis of the results reveals that re-ranking leads to the best performance for both languages. However, the introduction of N-grams (Shingles) technique significantly decreases the search engine's performance, nullifying the benefits of re-ranking and resulting in an even worse outcome compared to the best-performing configuration without this approach.

Figure 5 shows the interpolated precision-recall curves of the systems described in Table 8. This type of plot is useful to represent the trend of two different measures like precision and recall. In this way we can compare different systems to figure out which one is better than the others. System 5 intersects System 3 curve at recall 0.2. Hence, System 5 performs better at the high ranks and worse at the low ranks. Regarding other systems, the curves never intersect, so their performance are more clearly separated. Accordingly to the results reported in Table 8, System 4 is the best.

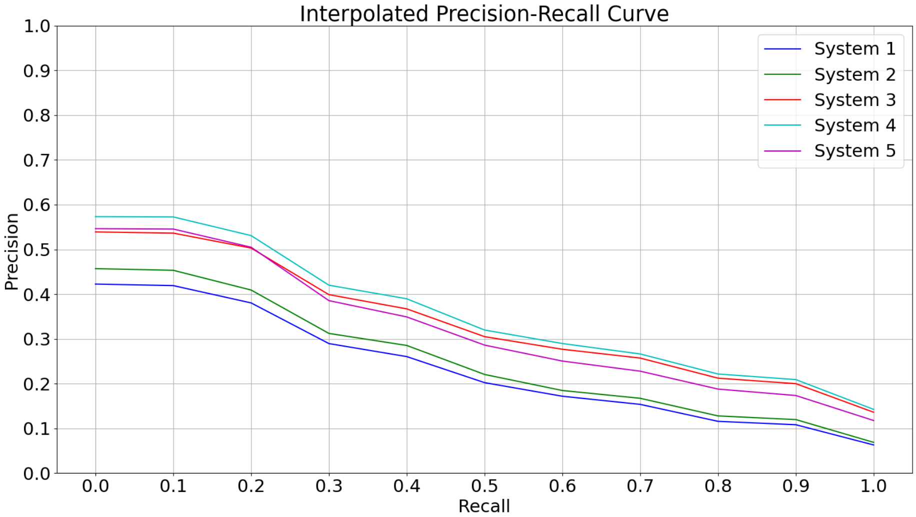


Figure 5: Interpolated Precision Recall curves on the training dataset

5.6. Test results

To conclude how well the submitted runs perform, we analyzed nDCG and MAP measures, and also performed Statistical Hypothesis Testing (SHT). The analysis was done for each of the five submitted runs, for both Short-term and Long-term collections. Short-term represents the collection from June 2023, while Long-term represents the collection from August 2023. This kind of analysis is essential as it shows how well each IR system performs. Moreover, it is particularly significant to compare these results since the purpose of the LongEval task is to find an IR system that can handle changes over time, meaning that we are interested in systems that have low measure drops over time. Table 9 shows the performance drop of the systems between June and August.

Table 9
Comparison between June and August

Label	System	nDCG		MAP	
		June	August	June	August
System 1	EN-Stop-SnowBall-Poss-Prox(50)	0.2938	0.2205	0.1564	0.1116
System 2	EN-Stop-SnowBall-Poss-Prox(50)-Reranking(200)	0.3039	0.2308	0.1686	0.1214
System 3	FR-Stop-FrenchLight-Elision-ICU-Prox(50)	0.3849	0.2855	0.2351	0.1615
System 4	FR-Stop-FrenchLight-Elision-ICU-Prox(50)-Reranking(150)	**0.3964**	**0.2942**	**0.2489**	**0.1709**
System 5	FR-Stop-FrenchLight-Elision-ICU-Shingles-Prox(50)-Reranking(150)	0.3701	0.2794	0.2204	0.1564

Additionally, SHT was performed to determine if the systems performance is statistically different or not. This is important to understand whether some configurations improve the system or not, for example, if the usage of reranking has a real effect on the performance or the mean increase is due only to the variance of the test.

In this section, we will further explain SHT, present the results for the previously mentioned metrics as well as for SHT, and provide our conclusions about them.

5.6.1. Statistical Hypothesis Testing

SHT is a type of statistical analysis used to estimate the relationship between statistical variables. Later in this section, the results produced by performing SHT on both Short-term and Long-term datasets will be shown and explained. SHT is important for determining in a scientifically valid way whether our systems are performing similarly or differently. In other words, we are interested in knowing if there is a statistical significant difference between them.

In order to perform SHT, the two mutually exclusive hypotheses H_0 and H_1 must be defined. H_0 is called *the null hypothesis* while H_1 is *the alternative hypothesis*. Beside these hypotheses a threshold α must be defined, representing a significance level. For example, $\alpha = 0.05$ means there is a 5% probability of wrongly declaring that the systems are different. SHT uses sample data to determine if H_0 can be rejected. If that is the case, it means that the alternative hypothesis H_1 is true [22].

In particular, we will use Two-Way Analysis of Variance (ANOVA2) as a statistical test. It examines the influence of two different variables which are, in our case, the systems and the topics. ANOVA2 is used to evaluate the difference between the means of more than two groups, which is useful in our case since we have 5 different IR systems. In ANOVA2 the hypotheses are as follows:

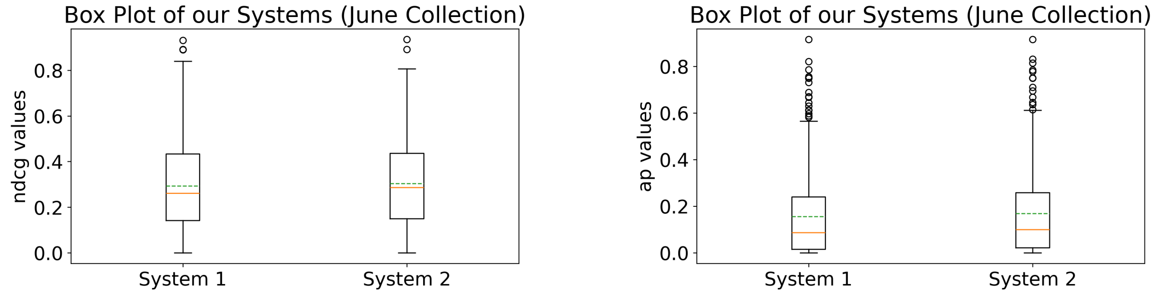
- H_0 - the means of all groups are equal,
- H_1 - at least 2 groups have different means [23].

5.6.2. Short-term

In this section, results for nDCG and AP measures, as well as for SHT, are presented for Short-term collection. Additionally, conclusions for the given results are provided.

Table 9 displays nDCG and MAP measures of the five submitted systems for the Short-term collection. It is evident that the best-performing system for the training data, based on both measures, remains consistent: FR-Stop-FrenchLight-Elision-ICU-Prox(50)-Reranking(150). However, there is a notable performance drop of 21.12% for nDCG and 19.16% for MAP. This drop was expected, as user queries and preferences have evolved over time. Notably, the systems with English configurations experience the largest drops. This phenomenon could be related to the automated translation process from French to English.

Based on the box plots in Figures 6 and 7, it is notable that the FR systems tend to have higher median values for both measures compared to the EN systems, indicating generally better performance. Additionally, the FR systems do not have as many outliers as the EN systems, which shows a more consistent performance.



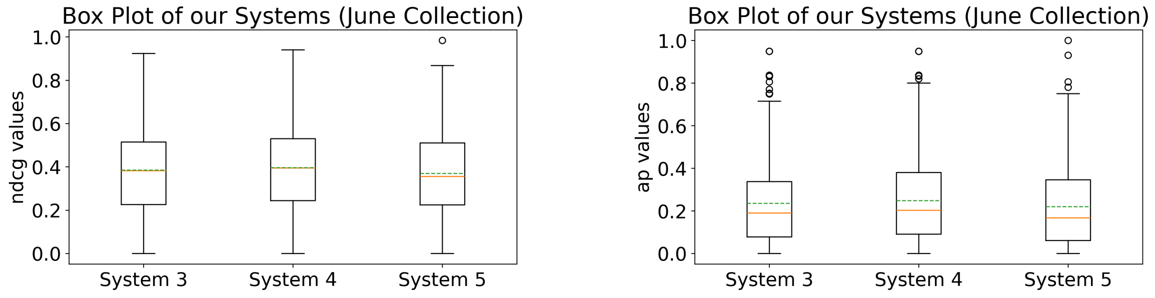
(a) nDCG

(b) AP

Figure 6: Box plots on Short-term collection for EN systems, the dashed lines represent the mean values and the solid ones the medians

Table 10 and Table 11 show the results of the ANOVA2 test.
The *SS* column shows the total variability for each source. Higher values indicate greater variability. *df* indicates how many degrees of freedom a source of variation has. For example, since the matrix of the scores contains a system for each column, and 5 systems are being compared, the columns will have 4 degrees of freedom.

The *MS* column is the average of the sum of squares (SS) for each source, calculated by dividing SS by its corresponding degrees of freedom (df). It represents the variance for each source.



(a) nDCG

(b) AP

Figure 7: Box plots on Short-term collection for FR systems, the dashed lines represent the mean values and the solid ones the medians

F shows the F-statistic, calculated as the ratio of the mean squares of the source to the mean squares of the error. It is used to determine if the observed variance between groups is significantly greater than the variance within groups.

Finally, the column we are most interested in, *Prob>F*, indicates the p-value associated with the F-statistic. A lower p-value (in our case less than 0.05) suggests that the differences between group means are statistically significant.

In this case, both Table 10 and Table 11, allow us to state that the 5 systems have different performance with a probability of being wrong very close to 0.

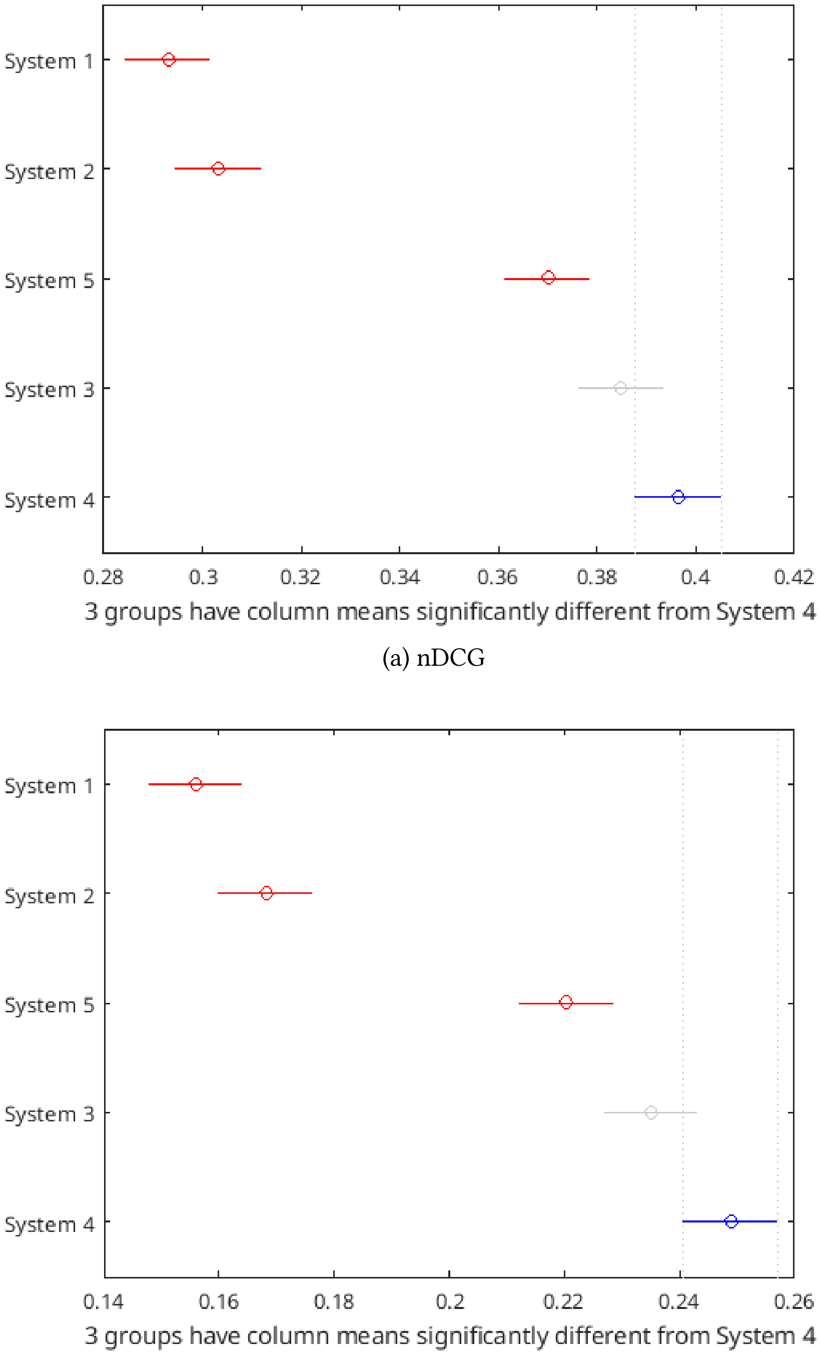
Table 10
Anova2 AP

Source	SS	df	MS	F	Prob>F
Systems	2.75807	4	0.68951	95.04877	0
Topics	63.14746	403	0.15669	21.59988	0
Error	11.69404	1612	0.00725		
Total	77.59958	2019			

Table 11
Anova2 nDCG

Source	SS	df	MS	F	Prob>F
Systems	3.71902	4	0.92975	113.77239	0
Topics	70.53095	403	0.17501	21.41620	0
Error	13.17338	1612	0.00817		
Total	87.42336	2019			

After discovering that the 5 systems are different, it is useful to compare systems pairwise in order to understand where the difference comes from. For this purpose, we will employ Tukey's Honest Significant Difference (HSD) test.



(a) nDCG

(b) AP

Figure 8: Plot of the differences in means across multiple groups, illustrating the variation within and between groups

Figure 8 shows a comparison between the mean of the different groups. It is useful to visualize this to understand if there is a real difference in the performance of the systems. For example, in both plots, System 1 was selected. This shows that there is a statistical difference between it and Systems 3, 4 and 5. For what concerns System 2, we can't say that it is different from System 1 with $\alpha = 0.05$.

The output of the test is reported in Table 12 and Table 13 for AP and nDCG, respectively. The p-values lower than $\alpha = 0.05$ are shown in bold, meaning that we refuse to accept the null hypothesis.

Table 12
Systems comparison AP

System A	System B	P-value
System 1	System 2	0.24863
System 1	System 3	**0**
System 1	System 4	**0**
System 1	System 5	**0**
System 2	System 3	**0**
System 2	System 4	**0**
System 2	System 5	**0**
System 3	System 4	0.14086
System 3	System 5	0.10402
System 4	System 5	**0.00001**

Table 13
Systems comparison nDCG

System A	System B	P-value
System 1	System 2	0.50142
System 1	System 3	**0**
System 1	System 4	**0**
System 1	System 5	**0**
System 2	System 3	**0**
System 2	System 4	**0**
System 2	System 5	**0**
System 3	System 4	0.36529
System 3	System 5	0.13639
System 4	System 5	**0.00033**

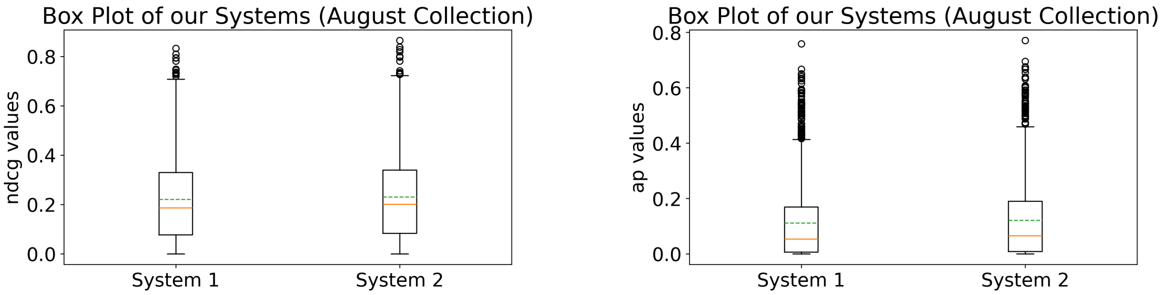
5.6.3. Long-term

In this section, results for nDCG and AP measures, as well as for SHT, are presented for Long term collection. Additionally, conclusions for given results are provided.

Table 9 shows the nDCG and AP scores of the five systems submitted for the Long-term collection. It's clear that the best-performing system, based on both measures, remains consistent: FR-Stop-FrenchLight-Elision-ICU-Prox(50)-Reranking(150). However, there's a significant performance decline of 41.45% for nDCG and 44.47% for MAP between training and Long-term data. This decline was expected, given the evolution of user queries and preferences over time.

Based on the box plots in Figures 9 and 10, we can make similar conclusions we made when comparing box plots for the Short-term collection - the FR systems showed better performance for both measures, compared to the EN systems. Moreover, the FR systems have fewer outliers than the EN systems. When comparing the box plots 9 and 10 for the Long-term collection with the box plots 6 and 7 for the Short-term collection, we can conclude the following: all five IR systems show a slight decrease in the median nDCG and AP values over time, which was expected. Another notable observation is that the box plots for the Long-term collection have more outliers than those for the Short-term collection. This behavior could be explained with various factors, such as the impact of different queries and documents and changes in user behavior over time.

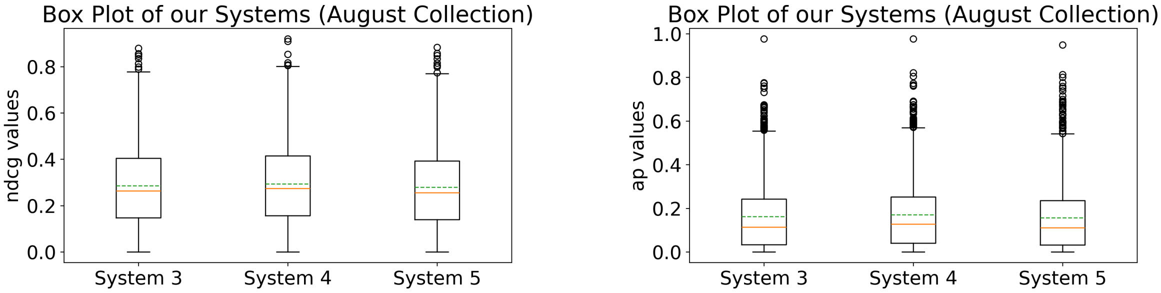
In the same fashion as what was done for the Short-term dataset in Section 5.6.2, we run the ANOVA2 test and the pairwise comparison using the HSD test. Table 14 and Table 15 show the results of the ANOVA2 test. The output of the systems comparison is reported in Table 16 and Table 17 for AP and nDCG, respectively. The p-values lower than $\alpha = 0.05$ are shown in bold, meaning that we refuse to accept the null hypothesis. One notable thing is that, on these datasets, all the systems, except for System 3 and System 5 are statistically different.



(a) nDCG

(b) AP

Figure 9: Box plots on Long-term collection for EN systems, the dashed lines represent the mean values and the solid ones the medians



(a) nDCG

(b) AP

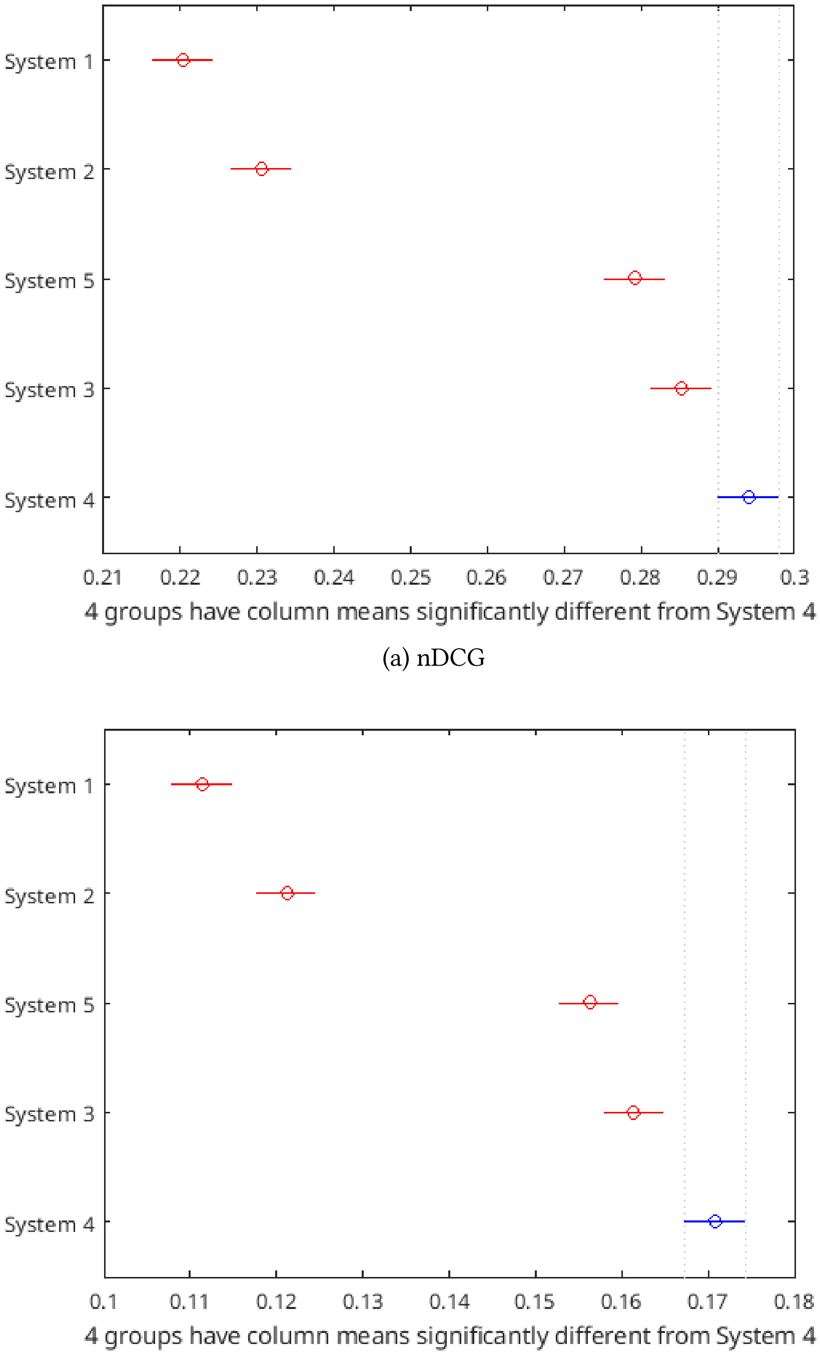
Figure 10: Box plots on Long-term collection for FR systems, the dashed lines represent the mean values and the solid ones the medians

Table 14
Anova2 AP

Source	SS	df	MS	F	Prob>F
Systems	4.17251	4	1.04312	212.17106	0
Topics	144.32133	1515	0.09526	19.37610	0
Error	29.79367	6060	0.00491		
Total	178.28752	7579			

Table 15
Anova2 nDCG

Source	SS	df	MS	F	Prob>F
Systems	6.96973	4	1.74243	269.51048	0
Topics	209.55709	1515	0.13832	21.39485	0
Error	39.17897	6060	0.00646		
Total	255.70580	7579			



(a) nDCG

(b) AP

Figure 11: Plot of the differences in means across multiple groups, illustrating the variation within and between groups

Table 16
Systems comparison AP

System A	System B	P-value
System 1	System 2	**0.00115**
System 1	System 3	**0**
System 1	System 4	**0**
System 1	System 5	**0**
System 2	System 3	**0**
System 2	System 4	**0**
System 2	System 5	**0**
System 3	System 4	**0.00221**
System 3	System 5	0.25629
System 4	System 5	**0**

Table 17
Systems comparison nDCG

System A	System B	P-value
System 1	System 2	**0.00419**
System 1	System 3	**0**
System 1	System 4	**0**
System 1	System 5	**0**
System 2	System 3	**0**
System 2	System 4	**0**
System 2	System 5	**0**
System 3	System 4	**0.02270**
System 3	System 5	0.22570
System 4	System 5	**0**

6. Conclusions and Future Work

In this section we summarize the main achievements and the conclusions we reached during the development of the SE. Firstly, using a basic configuration (with the standard tokenizer, the Snowball stemmer, and the lowercase filter), we observed better performance on the French dataset compared to the English dataset. We could explain this behavior with the fact that the translation to English is automated.

Secondly, the default performance shown in Section 5 is quite good, but after experimenting with different configurations, we managed to find better parameters. It's important to mention that we always used the appropriate stopword list, as we noticed an improvement in performance when utilizing them. Moreover, the SE performed better when using an appropriate stemmer, while the usage of synonyms had decremental effects on it. Another technique that brought significant progress was the usage of a reranker for both languages.

While analyzing performance drops and SHT results for both Short-term and Long-term collections, we arrived at the following conclusions. There is a significant difference between the systems with French and English configurations in terms of performance. The FR systems consistently outperform the EN systems in both datasets. As already mentioned, this is related to the automated translation from French to English and the fact that the language evolves over time [24]. This demonstrates that language-specific optimizations play a vital role in the effectiveness of retrieval systems. Moreover, systems with reranking generally perform better than non-reranking systems. Furthermore, the performance drop over time is evident, and it highlights the need of continuous updates to maintain performance over time.

Regarding future work, we could try to improve query expansion with the use of Large Language Models (LLMs), which have shown remarkable capabilities in the IR field, particularly in text understanding [25]. Indeed, in order to better perceive the user's intent and create a more efficient query, we could reformulate it. For this purpose, we could use a language model to rephrase the query and hopefully increase the performance of the SE. These models can grasp the context and meaning of text, enabling more accurate retrieval of relevant documents [26]. Another idea that could be beneficial to the performance of our system would be to use a sentence embedder model trained specifically on French data. This hypothetically could increase the boost in performance obtained with the use of the reranker even more.

References

[1] Sushilkumar Chavhan, M. Raghuwanshi, R. Dharmik, Information Retrieval using Machine learning for Ranking: A Review , https://iopscience.iop.org/article/10.1088/1742-6596/1913/1/012150/meta, 2021. [Online; accessed: 2024-05-22].

[2] R. Nogueira, W. Yang, K. Cho, J. Lin, Multi-stage document ranking with bert, 2019. URL: https://arxiv.org/abs/1910.14424. arXiv:1910.14424.

[3] 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.

[4] Apache Lucene Website, https://lucene.apache.org/, 2024. [Online; accessed: 2024-05-30].

[5] Jackson Github Page, https://github.com/FasterXML/jackson, 2024. [Online; accessed: 2024-06-01].

[6] S. Robertson, H. Zaragoza, et al., The probabilistic relevance framework: Bm25 and beyond, Foundations and Trends® in Information Retrieval 3 (2009) 333–389.

[7] LongEval 2024 Test Collection, https://doi.org/10.48436/xr350-79683, 2024. [Online; accessed: 2024-06-01].

[8] Lucene Query Parser Documentation, https://lucene.apache.org/core/9_10_0/queryparser/org/apache/lucene/queryparser/classic/package-summary.html, 2024. [Online; accessed: 2024-06-02].

[9] C. Fellbaum (Ed.), WordNet: An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.

[10] G. A. Miller, Wordnet: A lexical database for english, Communications of the ACM 38 (1995) 39–41.

[11] Nils Reimers, Iryna Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, https://arxiv.org/abs/1908.10084, 2019. [Online; accessed: 2024-05-05].

[12] all-mpnet-base-v2 Documentation, https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2024. [Online; accessed: 2024-06-05].

[13] M. Henderson, P. Budzianowski, I. Casanueva, S. Coope, D. Gerz, G. Kumar, N. Mrkšić, G. Spithourakis, P.-H. Su, I. Vulic, T.-H. Wen, A repository of conversational datasets, in: Proceedings of the Workshop on NLP for Conversational AI, 2019. URL: https://arxiv.org/abs/1904.06472, data available at github.com/PolyAI-LDN/conversational-datasets.

[14] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, Tie-Yan Liu, MPNet: Masked and Permuted Pre-training for Language Understanding , https://arxiv.org/abs/2004.09297, 2020. [Online; accessed: 2024-05-22].

[15] Flask Documentation, https://flask.palletsprojects.com/en/3.0.x/, 2024. [Online; accessed: 2024-06-01].

[16] W. Bruce Croft, Donald Metzler, Trevor Strohman, Search Engines - Information Retrieval in Practice, https://ciir.cs.umass.edu/irbook/, 2015. [Online; accessed: 2024-05-01].

[17] M. Fröbe, M. Wiegmann, N. Kolyada, B. Grahm, T. Elstner, F. Loebe, M. Hagen, B. Stein, M. Potthast, Continuous Integration for Reproducible Shared Tasks with TIRA.io, in: J. Kamps, L. Goeuriot, F. Crestani, M. Maistro, H. Joho, B. Davis, C. Gurrin, U. Kruschwitz, A. Caputo (Eds.), Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023), Lecture Notes in Computer Science, Springer, Berlin Heidelberg New York, 2023, pp. 236–241. doi:10.1007/978-3-031-28241-6_20.

[18] M. Fröbe, J. Reimer, S. MacAvaney, N. Deckers, S. Reich, J. Bevendorff, B. Stein, M. Hagen, M. Potthast, The Information Retrieval Experiment Platform, in: H. Chen, W. E. Duh, H. Huang, M. P. Kato, J. Mothe, B. Poblete (Eds.), 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2023), ACM, 2023, pp. 2826–2836. doi:10.1145/3539618.3591888.

[19] C. Macdonald, N. Tonellotto, S. MacAvaney, I. Ounis, Pyterrier: Declarative experimentation in python from bm25 to dense retrieval, in: Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM 2021), 2021, pp. 4526–4533.

[20] Aparna Dhinakaran, Demystifying NDCG, https://towardsdatascience.com/demystifying-ndcg-bee3be58cfe0, 2023. [Online; accessed: 2024-05-02].

[21] Ren Jie Tan, Breaking Down Mean Average Precision (mAP), https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52, 2019. [Online; accessed: 2024-05-02].

[22] Christina Majaski, Hypothesis Testing: 4 Steps and Example, https://www.investopedia.com/terms/h/hypothesistesting.asp, 2024. [Online; accessed: 2024-05-15].

[23] Will Kenton, What Is Analysis of Variance (ANOVA)?, https://www.investopedia.com/terms/a/anova.asp, 2024. [Online; accessed: 2024-05-15].

[24] Rabab Alkhalifa, Elena Kochkina, Arkaitz Zubiaga, Building for tomorrow: Assessing the temporal persistence of text classifiers, https://arxiv.org/abs/2205.05435, 2022. [Online; accessed: 2024-05-22].

[25] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, Ji-Rong Wen , Large Language Models for Information Retrieval: A Survey, https://arxiv.org/pdf/2308.07107, 2024. [Online; accessed: 2024-05-04].

[26] Vishal Gupta, Ashutosh Dixit, Shilpa Sethi, An Improved Sentence Embeddings based Information Retrieval Technique using Query Reformulation, https://ieeexplore.ieee.org/document/10141788, 2023. [Online; accessed: 2024-05-04].