SEUPD@CLEF: Team QEVALS on Information Retrieval Adapted to the Temporal Evolution of Web Documents

Notebook for the LongEval Lab at CLEF 2023

Enrico D'Alberton[1], Saverio Fincato[1], Vaidas Lenartavicius[1], Laura Pallante[1], Yijian Qiu[1] and Nicola Ferro[1]

[1]*University of Padua, Italy*

Abstract

This report presents the work conducted by our team for LongEval-Retrieval Task 1 [1] in CLEF 2023 [2]. The primary objective of this task is to develop an information retrieval system that can effectively adapt to the temporal evolution of Web documents. Using the Longeval Websearch collection provided by the commercial search engine Qwant[3], our team has built a retrieval system that addresses the challenges posed by the changing nature of Web documents and user search preferences. This paper discusses our approach to the subtasks of short-term persistence and long-term persistence, as well as the evaluation of our retrieval system's performance.

Keywords

CLEF 2023, LongEval-Retrieval, Information Retrieval, Temporal Evolution, Search Engines, Short-term Persistence, Long-term Persistence

1. Introduction

Nowadays, the advent of the internet has led to an exponential growth of the information available online and this brought a more challenging way to find relevant and accurate information. This is the environment where search engines play a vital role that allows us to access the correct information quickly and efficiently just with a few clicks. Information retrieval systems are more crucial than ever, influencing several fields such as healthcare, business and mainly education.
Our objective is to investigate the subject of information retrieval in search engines and its adaptability to changes over time. The necessity to create a retrieval system that can successfully handle the dynamic nature of the web is what stimulates this research's development.
Our team, QEVALS, is participating in this challenge as a student group project conducted in the Search Engines course a.y. 2022/23 at the Computer Engineering master's degree at the

CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece
enrico.dalberton@studenti.unipd.it (E. D'Alberton); saverio.fincato@studenti.unipd.it (S. Fincato); vaidas.lenartavicius@studenti.unipd.it (V. Lenartavicius); laura.pallante@studenti.unipd.it (L. Pallante); yijian.qiu@studenti.unipd.it (Y. Qiu); ferro@dei.unipd.it (N. Ferro)
http://www.dei.unipd.it/~ferro/ (N. Ferro)
0000-0001-9219-6239 (N. Ferro)

University of Padua. We'll be working on the Longeval [4] Websearch collection, a sizable collection of data made up of web pages, user interactions, and queries made available by Qwant, a privacy-focused French search engine.
This project helped develop our understanding of the information retrieval systems used in the context of a web search engine. Specifically, we applied ourselves to the task of processing queries and documents in order to get the best possible ranking results to provide users with the most relevant information. The paper is organized as follows: Section 2 describes our approach; Section 3 explains our experimental setup; Section 4 discusses our main findings; finally, Section 5 draws some conclusions and outlooks for future work.

2. Methodology

The development process was iterative and based heavily on discussions between the group members regarding the subjects covered during the lectures. However, our first results were significantly below the baseline set by the organizers of the LongEval task.

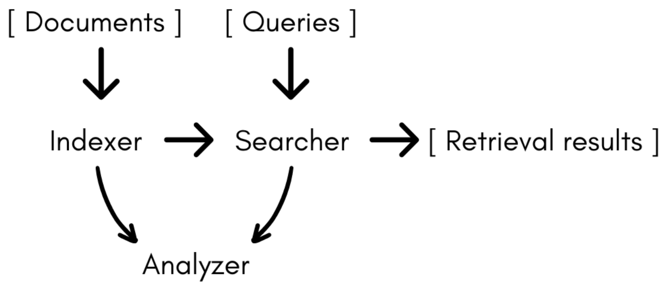


Figure 1: Scheme of the project's workflow

2.1. Analyzer

The initial implementations of the classes were very simple and we started adding features to them day by day. The main purpose of an analyzer in an IR model is to pre-process the input data in a way that reduces the complexity of the document representation, while still retaining the relevant information necessary for accurate retrieval. The final models we submitted, present a tokenizer implemented using `LetterTokenizer` class from Apache Lucene [5] and it simply divides text at non-letters. Subsequently, we applied a `LowerCaseFilter` and an `ElisionFilter`, which, respectively, normalize token text to lowercase and remove elisions from a token. This last filter has been used specifically for our case since the dataset provided by LongEval is in French. We have also applied a `ASCIIFoldingFilter` to convert alphabetic, numeric, and symbolic Unicode characters that are not in the first 127 ASCII characters (the "Basic Latin" Unicode block) into their ASCII equivalents.
One of the tools that had the biggest impact on our system's performance was the implementation of a `StopFilter`. As the name suggests, it is used to remove stopwords (very frequent

words that contain little information about the contents of a document or a query) from a stream of text. While testing different stoplists and experimenting with query boosting some parts of the queries, we discovered that for some stoplists, individually boosting query tokens that were non-stopwords would improve system performance, even though the queries were subject to a `StopFilter` later down the pipeline. We tried using the same stoplist for our query boosting and our `StopFilter`, as well as different ones and found that the best combination varied on a case-by-case basis. This particular type of query boosting was effective only on less-performing stoplists, thus it isn't included in the final system. These are the results compared:

Table 1
MAP values

	CFW	Google	Iso	Rank	Savoy	NoStop
NoFilter	0.194	**0.2082**	0.1932	**0.2076**	0.1947	**0.2018**
OwnFilter	**0.2024**	0.2011	**0.2011**	0.2015	0.2018	0.2018
FST	0.2021	0.201	**0.2011**	0.2007	**0.2025**	

Table 2
nDCG values

	CFW	Google	Iso	Rank	Savoy	NoStop
NoFilter	0.3256	**0.3432**	0.3248	**0.3422**	0.326	**0.3376**
OwnFilter	0.3371	0.3365	0.3361	0.3369	0.3367	0.3376
FST	**0.3374**	0.3359	**0.3364**	0.3364	**0.3376**	

Table 3
P@10 values

	CFW	Google	Iso	Rank	Savoy	NoStop
NoFilter	**0.1265**	**0.133**	**0.1262**	**0.1316**	**0.1269**	**0.1257**
OwnFilter	0.1255	0.1256	0.1247	0.125	0.1253	0.1257
FST	0.126	0.1253	0.1252	0.1247	0.1262	

Table 4
Recall values

	CFW	Google	Iso	Rank	Savoy	NoStop
NoFilter	0.672	**0.6899**	0.672	**0.6867**	0.6718	**0.6791**
OwnFilter	0.678	0.6804	0.678	0.679	0.6774	0.6791
FST	**0.6796**	0.6778	**0.6791**	0.6779	**0.6795**	

The lists we used are:

- **CountWordsFree [6]**: CountWordsFree is a web-based service for content writers, web developers, and professionals who provide search ranking optimization services such as the list of stopwords we have used;
- **Google Stopwords [7]**: set of common words that are filtered out or ignored by Google's search engine algorithms when processing search queries;
- **Stopwords ISO [8]**: collection of stop words for various languages that are commonly used in natural language processing (NLP) tasks;
- **Ranks.nl [9]**: Dutch website that provides various online marketing services and tools;
- **Savoy Stopwords [10]**: it is a standard library that provides a collection of robust, high-performance libraries for mathematics, statistics, data processing, streams, and more and includes many of the utilities you would expect from a standard library;
- **NoStop**: it is simply an empty list, to evaluate if the filter was enhancing the performances or not.

As we can see from the obtained values, the Google stopwords list is the one that performs best, without stoplist based query boosting. It is worth noting that it is the shortest list that was tested.

2.1.1. Tested but unused filters

The systems submitted by our group are the results of numerous trials and runs, and unfortunately many of our attempted approaches proved themselves to be ineffective or unfeasible.
In our first prototypes, a `LengthFilter` was used to remove the words with a number of characters below and over certain thresholds. Initially it incremented the model's scores, but upon further refinements of other parts of the system it started having a negative impact on performance.
A few different configurations of a `ShingleFilter` were also attempted to create tokens from overlapping sequences of n words (token n-grams) from a token stream. We found they weren't suited to the LongEval task.
Multiple attempts were made to integrate an NLP (Natural Language Processing) library into our system. It could be used for many tasks, such as tokenization, sentence segmentation, part-of-speech tagging and named entity extraction. Specifically, we focused on two different libraries: OpenNLP [11] and CoreNLP [12]. We tried running many configurations of both libraries, however for a dataset as large as LongEval's they were requiring computational power well beyond what we had access to, therefore we were forced to discard the idea.

2.2. Indexer

The Indexer is a class responsible for creating an index of the terms in a collection of documents. The purpose of the indexer is to speed up the retrieval of relevant documents in response to user queries. It creates an internal data structure, the 'index', that allows quicker access to data. An indexer reads through the text data, identifies important and searchable entities, and constructs an index of those entities and their location in the data.

Our custom indexer, called `LongEvalIndexer`, is an update of the initial indexer design user for the Tipster collection discussed during the lectures. The latest version of it, recursively visits each file in the specified document directory and extracts the contained text data through the Parser class. Each document is parsed and javascript or PHP code is removed through the TextFilter class. Subsequently, each document is stored in the index and divided into the following fields:

- **IDField:** this field stores the unique identifier associated with each document;
- **BodyField:** the body of the document is indexed and stored in this field after being processed by the parser. The field type is defined using a `FieldType` object, which is configured to store the term vectors, positions, and offsets;
- **LinkField:** the link corresponding to the document is extracted from the URLs file provided in the LongEval dataset and is processed to remove specific characters in order to obtain the words contained in the link. It is then added to the document using a custom `LinkField`, which is configured to store the term vectors, positions, and offsets.

2.3. Searcher

The Searcher is a class used to represent the user or the query that seeks information from a collection of documents, so its main function is to express the information need of the user in the form of a query. This is the last class to be used in the information retrieval process since to perform research on the documents they have to be indexed first. The final version of this class takes as input a set of queries from the set provided by LongEval and it builds a `BoostQuery` object from each query. The `BoostQuery` class allows to give a boost to the wrapped query. Boost values that are less than one will give less importance to this query compared to other ones while values that are greater than one will give more importance to the scores returned by this query. Combined with this class, we exploited the `MultiFieldQueryParser` class, which is used to associate queries and documents with multiple fields, also defining the weight they will have in the search. In our case, the fields we researched were "body" and "link". Using different weights, we noticed that the weights given to these fields were crucial for the model's performance.

Other approaches for the implementation of this class were attempted but did not result in performance gains. We tested two types of query expansion techniques, one based on the top relevant documents for each query and another one developed with the help of `gpt-3.5-turbo` model by OpenAI [13]. The functioning of the first one was based on an algorithm that found the most relevant words in the top ten retrieved documents for each query and then it added these words to the main query. After that, a search on the collection was repeated with the expanded query. Unfortunately, this technique proved ineffective, as it significantly lowered

the accuracy of the system. This due to the low accuracy rate that the system can achieve even in the best-ranked dopuments. This led to adding more noise words than relevant and thus worsening the search, forcing us to abandon this approach.
The second query expansion we tested made use of an NLP model developed by OpenAI. The idea was to give the model a single query as input and ask it to return an expanded one. The first problem we encountered was linked to the number of calls we could do, by default OpenAI limits users to 3 calls per minute, but each run was about 700 queries. In order to avoid this bottleneck, we formed a prompt to expand 100 queries in each call. Unfortunately, the model was not particularly accurate and sometimes it diverged too much, expanding the queries incorrectly. For this reason, we had to abandon this idea too, but we believe that with some refinements it could be a good improvement for a future model.

2.4. Similarity

Similarity function is a fundamental component used to measure the similarity, or relatedness, between queries and documents. The function returns a score that reflects how well the document matches the query. During the development process the BM25 similarity model with default parameters was used as a baseline. The systems our team is submitting for the LongEval task have various configurations of similarity models in order to test how they perform with separate but similar datasets and study any trends that emerge. The models being tested are described in the Results 4 section.

3. Experimental Setup

3.1. Data description

The data used in the project are the evaluation collection provided for task 1, "LongEval-Retrieval", consisting of collections of web documents and queries provided by Qwant, a search engine. Both documents and queries were collected in French and then automatically translated into English. This scheme represents the collection process: The data can be described as follows:

3.1.1. Documents

The collection includes relevant documents that are selected to be retrieved for each query and potentially irrelevant documents randomly sampled from the Qwant index to better represent the nature of a Web test collection.

- **Train data:** consists of 1,570,734 web pages, acquired during June 2022. They can be downloaded from the Lindat/Clarin website.

- **Test data:** consist of 1,593,376 documents and 882 queries, collected over July 2022, for the short-term persistence sub-task and 1,081,334 documents and 923 queries, collected over September 2022, for the long-term persistence sub-task. Both can be downloaded from the Lindat/Clarin website.

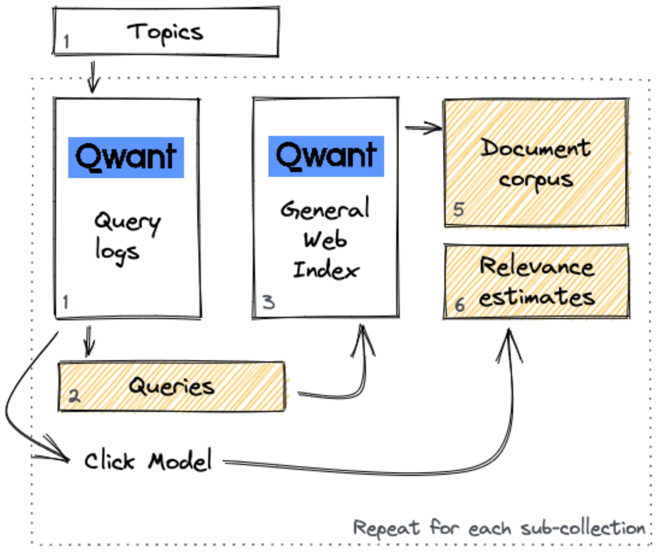


Figure 2: Scheme of the collection process [1]

3.1.2. Topics

The queries are extracted from Qwant's search logs, based on a set of selected topics, but exact details regarding them were not made available to us.

3.1.3. Queries

The queries are extracted from Qwant's search logs, based on a set of selected topics. The query set was created in French and was automatically translated into English.

3.2. Evaluation Measures

The evaluation tools used during development are **trec_eval** [14], an executable that through qrels allows us to obtain measures of precision, accuracy, etc., and **Luke** [15], a Lucene GUI that allows us to look inside the index, to check the health and consistency of the indexes.
The main evaluation measures used to check the system during the various stages of development are:

- **nDCG** or Normalized Discounted Cumulated Gain: it measures the effectiveness of a retrieval system by considering the relevance of documents and their positions in the ranked list. In our case, we used an evaluation depth of 5 (nDCG@5). To compute it is necessary to normalize the score by dividing nDCG by the iDCG (Ideal Discounted Cumulated Gain). These are the formulas to compute them:

$$DCG@k = \sum_{n=1}^{k} \frac{rel_n}{max(1, log_b(n))} \tag{1}$$

$$nDCG@k = \frac{DCG@k}{iDCG@k} \tag{2}$$

- **MAP** or Mean Average Precision: this is the most widely used metric in IR and it measures the average precision across all relevant documents in the retrieval set. It is given by:

$$\text{AP} = \sum_{n} (R_n - R_{n-1}) P_n \tag{3}$$

  where R_n and P_n are the precision and recall at the n^{th} threshold.

3.3. Repository

The git repository containing the source code of the system is available in the repository at the link seupd2223-qevals (https://bitbucket.org/upd-dei-stud-prj/seupd2223-qevals).

3.4. Hardware components

- Saverio's PC:
  - CPU: Intel® Core™ i5-8600K 3.60GHz × 6
  - GPU: NVIDIA GeForce GTX 1060 6GB
  - RAM: 16 GB DDR4
  - SO: Ubuntu 22.04.2 LTS

- Enrico's PC (laptop)
  - CPU: Intel Core i7-8550U 1.8 GHz Base, 4.0 GHz Turbo
  - GPU: NVIDIA GeForce MX250 2GB GDDR5
  - RAM: 16 GiB DDR4
  - SO: Pop!_OS 22.04 LTS

4. Results and Discussion

The numerous runs that were executed on the LongEval training dataset provided us with good baseline analyzer and searcher configurations that we could use to compare how different similarity models performed on the LongEval task. Specifically, we learned that:

- Better results were obtained working on the dataset containing documents in the original language compared to the English-translated one.
- In the case of French queries and documents the best-performing tokenization is obtained through Lucene's LetterTokenizer.
- Stop Lists can improve performance, but most available lists tend to over-filter the data to the point of performing worse than using no filter at all.
- In case of stop list over-filtering it is possible to regain some performance by query boosting tokens in the query that are specifically not in a stop list, even though the stop list tokens get filtered later in the pipeline.

- The best performing Stop Lists had at most 200 words. Specifically, we chose the Google stop list [7] as it lightly edged out the Ranks stop list [9].
- The use of stemmers had little effect, but seemed to reduce overall performance. We tested LightFrenchStemmer and MinimalFrenchStemmer, available on Apache Lucene, and chose to proceed without using either.
- Adding more filters (other than the LowercaseFilter, the ElisionFilter and the ASCIIFoldingFilter) such as the LengthFilter, in an attempt to exclude less useful terms, led to worse performance.
- The use of word n-gram tokens was ineffective, possibly because such short queries are often not semantically coherent sentences, so they would not match the contents of the longer documents.
- Our implementations of Query Expansion did not perform well, though we are uncertain whether the LongEval dataset is ill-suited to this approach or our implementations had inherent issues.
- Given the very short format of the queries adding the link field of the document to the index led to notable performance improvements.

Using the configuration described above five different similarity models were tested:

- **QEVALS_BM25DFLT**: BM25 similarity model with default parameters (k1 = 1.2, b = 0.75).
- **QEVALS_BM25CSTM**: BM25 similarity model with the best-performing parameters that were found (k1= 1.2, b = 0.9).
- **QEVALS_LMDirichlet**: Language model based similarity with Bayesian smoothing using Dirichlet priors.
- **QEVALS_DFR**: Probabilistic model that measures the divergence from randomness. The three components chosen were: Geometric approximation of Bose-Einstein, Laplace's law of succession and Uniform distribution of term frequency.
- **QEVALS_IB**: Information-based model. The three components chosen were: Log-logistic probabilistic distribution, Total Term Frequency Lambda and Uniform distribution of term frequency.

4.1. Training set

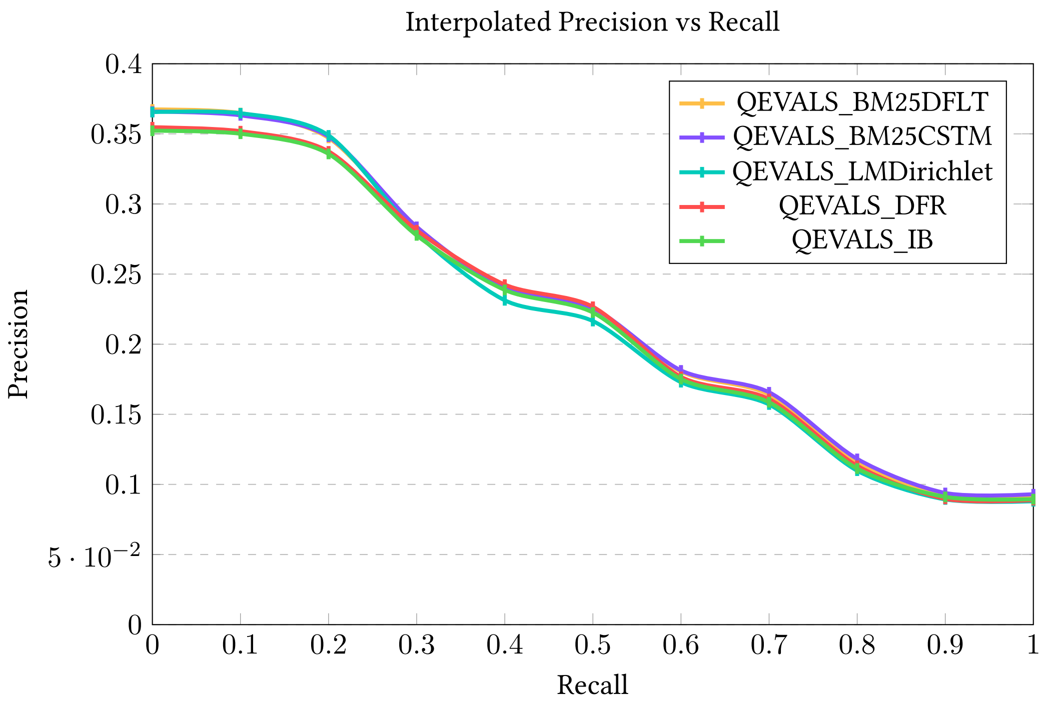
Following is the summary table reporting the MAP and nDCG values obtained in the training set for each run:

Table 5
Summary table with MAP and nDCG (training set)

	MAP	nDCG
QEVALS_BM25DFLT	0.2078	0.3433
QEVALS_BM25CSTM	0.2082	0.3432
QEVALS_LMDirichlet	0.2042	0.3417
QEVALS_DFR	0.2035	0.3377
QEVALS_IB	0.2024	0.3383

As reported, the values of each run are similar, in fact the variances for both sets of measures are very low. This is due to the fact that the changes across the models are not so relevant, indeed the main structure is almost the same.

Here instead, there is the Interpolated Precision vs Recall graph for the five tested similarity models:



The graph shows that the performances of the chosen similarity models are quite close to each other for the LongEval training dataset. In particular, BM25 performs best overall, and changes to the default parameters can result in further (though limited) gains in performance. The language model is very slightly better than BM25 at low recall, but drops off more steeply at around 30% Recall, while the statistical models have comparable performance to BM25 at most points, but lag behind at low recall.

4.2. Test Set

This section aims to provide a comprehensive analysis of the performance of the five models submitted for evaluation. Various metrics are used to evaluate the effectiveness and efficiency of each model. These metrics include average precision(MAP), precision(P@10), normalized discounted cumulated gain (nDCG) and recall (Recall_1000).
First, we present three tables, each corresponding to one of the test sets. These tables compare the performance of each run based on the metrics listed above.
Furthermore, we reported the boxplots calculated from the average precision (AP) respectively for short and long term. These graphical representations are intended to provide a detailed overview of the variability and distortion of the system's accuracy values.
Next, there are the comparative graphs of the MAP values obtained with the Tukey's HSD of the runs for short and long term. This will offer a comparison of the mean performance of the systems, highlighting their respective strengths and weaknesses.
Subsequently, we reported the ANOVA tables (one-way) respectively for long and short term. These tables show how the sum of squares are distributed according to source of variation, and hence the mean sum of squares.
Finally, we have plotted the topics performance at 5 for short and long term.

Table 6
Run performance

	Heldout			
	MAP	P@10	nDCG	Recall
QEVALS_BM25CSTM	0.1975	0.1293	0.3545	0.7688
QEVALS_BM25DFLT	0.2017	0.1268	0.3584	**0.7749**
QEVALS_DFR	0.2015	0.1317	0.3556	0.7635
QEVALS_LMDirichlet	0.1879	0.1171	0.3461	0.7738
QEVALS_IB	**0.2108**	**0.1329**	**0.3597**	0.7635

Table 7
Run performance

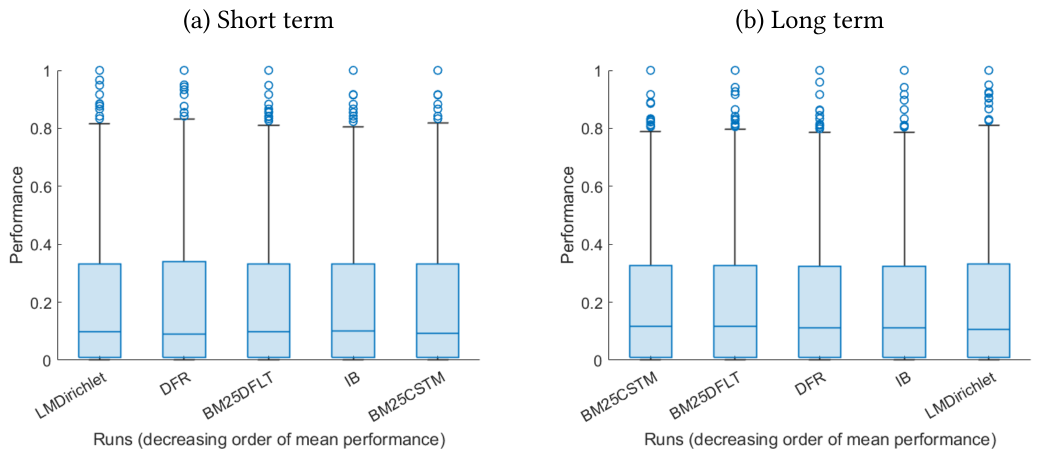
	Short			
	MAP	P@10	nDCG	Recall
QEVALS_BM25CSTM	0.1998	0.1303	0.334	**0.6874**
QEVALS_BM25DFLT	**0.2039**	**0.131**	0.3376	0.6872
QEVALS_DFR	0.1957	0.1295	0.3304	0.6849
QEVALS_LMDirichlet	0.2027	0.1254	**0.3392**	0.6871
QEVALS_IB	0.1966	0.1296	0.3321	**0.6874**

Table 8
Run performance

	Long			
	MAP	P@10	nDCG	Recall
QEVALS_BM25CSTM	0.2007	0.1389	0.3397	0.6797
QEVALS_BM25DFLT	**0.2015**	**0.1395**	**0.3404**	0.6803
QEVALS_DFR	0.1981	0.137	0.3372	0.6801
QEVALS_LMDirichlet	0.195	0.1329	0.335	0.6778
QEVALS_IB	0.1987	0.1371	0.3383	**0.6804**

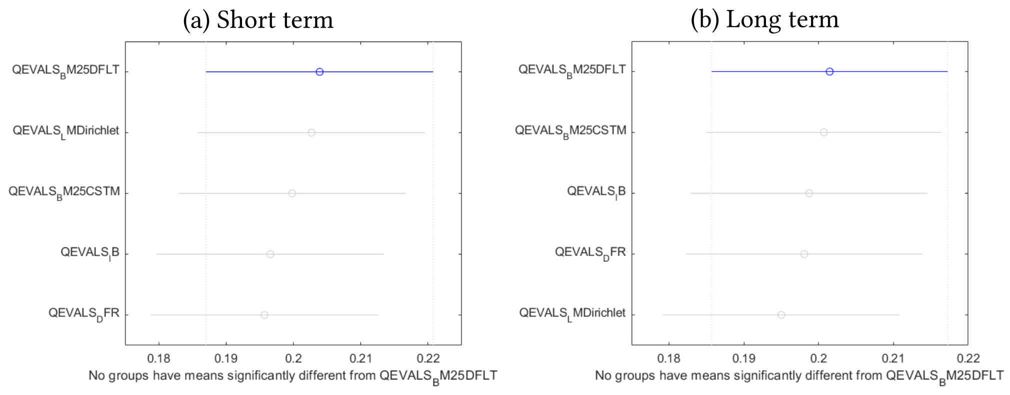
In **Tables 6, 7, 8** the observed values of the different "runs" of the system are similar, without significant deviations, and we can observe that the MAP values in both tables remain around 20%. It is interesting to note that the QEVALS_IB run performed well in the heldout set, obtaining the best results (map: 0.2108, p@10: 0.1329, nDCG: 0.3597), indeed it did not scored as expected in the Short and Long Test datasets. The values obtained from run QEVALS_BM25DFLT in the short and long test set seem slightly better than the others.

Figure 3: Precision BoxPlots



From the boxplots in **Figure 3** we can see that the accuracy values obtained from the runs are very similar, also we see that QEVALS_LMDirichlet and QEVALS_BM25CSTM seem to perform well in one test set and less in the other, instead QEVALS_DFR and QEVALS_BM25DFLT seem to maintain performance in both.

Figure 4: Runs comparison



From the **Figure 4**, even if there are not relevant differences in performances, as already confirmed from the previous analysis, we can highlight that the best run is QEVALS_BM25DFLT.

Table 9
ANOVA2 (two-way) Short Term

Source	SS	df	MS	F	Prob>F
Columns	0.0399	4	0.0099	0.1756	0.9510
Rows	36.0256	145	0.2484	4.3667	1.824e-54
Interaction	2.2227	580	0.0038	0.0673	1
Error	166.1379	2920	0.0568		
Total	204.4262	3649			

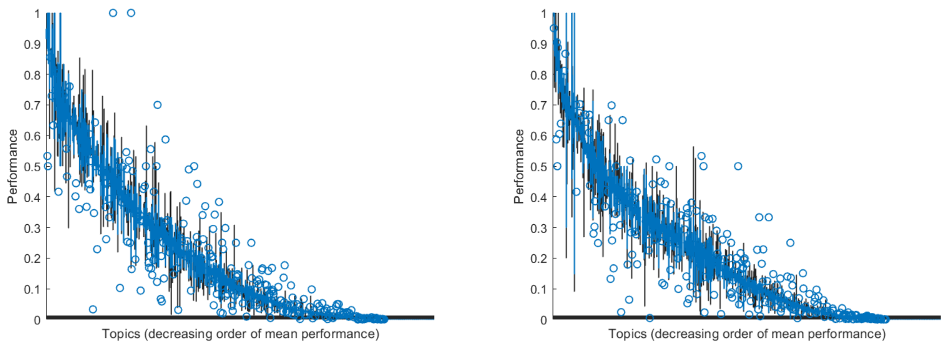
Table 10
ANOVA2 (two-way) Long Term

Source	SS	df	MS	F	Prob>F
Columns	0.0203	4	0.0050	0.1024	0.9816
Rows	44.0663	153	0.28801	5.8022	5.2730e-88
Interaction	2.1034	612	0.0034	0.0692	1
Error	152.8868	3080	0.0496		
Total	199.0770	3849			

From the analysis of the two-way ANOVA, visible in **Table 9** and **Table 10**, it is confirmed how there is no substantial difference between the runs. A low F-value statistic suggests that the variation between the groups is relatively small compared to the variation within them; in addition, a high F-value probability indicates that the probability that the observed differences are due to chance is very high. This further confirms that the differences between runs tested are not statistically significant.

Figure 5: Topics performance

(a) Short term (b) Long term



Similarly to the training dataset, the different runs on the short-term and long-term datasets manifested significant consistency, with similar MAP, nDCG, and Recall values. This indicates a relative stability of system performance and shows the capacity of adaptation to the temporal evolution of the documents.

As expected, the overall performance of the runs showed no significant variation between our training baseline and the short-term and long-term document sets. **Figure** 5, in particular, shows that our system's performance is virtually identical for the entire range of topics of the short-term and long-term datasets. The capability to have consistent performance regardless of the contents of a dataset can be considered an advantage of a system such as ours, which does not use past data to train a model for future applications. One notable aspect that we can infer from the test data is that the fine-tuning of the parameters of the BM25 similarity model is quite dependent on the available data. In fact, for both short-term and long-term dataset test runs, the default BM25 model performed slightly better than our fine-tuned one.
However, in general, the overall performances don't change drastically. Indeed, the values of the measurements differ at most $\sim 2\%$. We can conclude that for a task such as LongEval the similarity model choice alone has little impact on the overall performance of the system.

Finally, QEVALS_BM25DFLT (the default configuration of the BM25 similarity model) proved to be the best performing run, achieving the highest MAP, nDCG and Recall values among the different runs evaluated. Our conclusion is that this system is the most suited to the task out of the ones we tested.

5. Conclusions and Future Work

While our systems do not perform much better than the baseline provided by LongEvals, the process to arrive at the systems we are submitting offered us an opportunity to acquire experience and knowledge in dealing with information retrieval systems. Many of our approaches did not perform as hoped, however thanks to our numerous trials we are much better equipped to try new approaches to information retrieval in the future. Moreover, we are aware of many possible enhancements to the project.

More studies on the datasets

Our first challenge was the language. At a baseline, performance is better on the dataset in the original language, so our team decided to focus more on the French dataset early on. Our initial attempts on the English dataset seemed less promising, but it's possible that we would have had more success implementing the more complex systems for it.

5.1. Usage of NLP

The use of an NLP library such as OpenNLP or CoreNLP would have helped us take a big step forward. We learned that using Natural Language Processing technology to implement the whole model is probably unfeasible since it would be too expensive from a computational point of view, but we believe that a part of part-of-speech tagging or named entity extraction would have been helpful. Therefore, if we had a chance to run the project on more powerful computers, and sufficient time to build efficient parallelization algorithms, we believe that the model could gain better results and performances

5.2. Query expansion techniques

Another important aspect we were not able to implement properly was a query expansion technique, as discussed in the Methodology 4 section. There are many algorithms that could be used and some of them are already present in some libraries. The main problem was that our dataset was in French and we have not been able to find a good dictionary in order to expand the queries. So, we think that a query expansion technique with a proper dictionary could be a good addition to the project. Moreover, the two algorithms we tried to implement could be improved. In fact, the first one, which expands the query using the top retrieved documents' words, may already be in a working state, but in order for it to work well, the rest of the system must provide precise enough results at low recall values. This is due to the fact that a better base model leads to more relevant documents and so, to more relevant words with which to expand the query. Due to low precision at low recall we were adding noise to the initial query that, in the end, was performing worse than the original one.

The query expansion technique which was using the `gpt-3.5-turbo model`, was not a success either, however, it must be noted that compared to our other approaches it is the one we worked on the least because of time limitations and OpenAI's API access quota limitations for non-paying users. The problem our team ran into was the unpredictability of the answers: the model was asked to expand each query to five words, but in some instances it expanded queries to up to fifteen words, adding useless noise. For such a system to work, some additional experience in prompt formulation for information retrieval purposes would likely bring much better results.

References

[1] CLEF, Longeval, 2023. URL: https://clef-longeval.github.io/.
[2] CLEF, Clef 2023, 2022. URL: http://clef2023.clef-initiative.eu/index.php.
[3] Qwant, About qwant, 2011. URL: https://about.qwant.com/.
[4] R. Alkhalifa, I. Bilal, H. Borkakoty, J. Camacho-Collados, R. Deveaud, A. El-Ebshihy, L. Espinosa-Anke, G. Gonzalez-Saez, P. Galuscakova, L. Goeuriot, E. Kochkina, M. Liakata, D. Loureiro, H. T. Madabushi, P. Mulhem, F. Piroi, M. Popel, C. Servan, A. Zubiaga, Overview of the clef-2023 longeval lab on longitudinal evaluation of model performance, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2023), Lecture Notes in Computer Science (LNCS), Springer, Thessaliniki, Greece, 2023.
[5] Apache, Apache lucene, 2000. URL: https://lucene.apache.org/.
[6] CountWordsFree, Countwordsfree french stopwords, 2023. URL: https://countwordsfree.com/stopwords/french.
[7] Google french stopwords, 2023. URL: https://meta.wikimedia.org/wiki/Stop_word_list/google_stop_word_list#French.
[8] Stopwords-Iso, Stopwords-iso/stopwords-fr: French stopwords collection, 2023. URL: https://github.com/stopwords-iso/stopwords-fr.
[9] Ranks.nl french stopwords, 2023. URL: https://www.ranks.nl/stopwords/french.
[10] Stdlib-Js, Stdlib-js/datasets-savoy-stopwords-fr: A list of french stop words., 2023. URL: https://github.com/stdlib-js/datasets-savoy-stopwords-fr.
[11] Apache, Opennlp, 2004. URL: https://opennlp.apache.org/.
[12] S. N. Group, corenlp, 2010. URL: https://stanfordnlp.github.io/CoreNLP/.
[13] OpenAI, Models - openai, 2023. URL: https://platform.openai.com/docs/models/gpt-3-5.
[14] NIST, trec_eval, 2023. URL: https://trec.nist.gov/trec_eval/.
[15] Apache, Luke, 2009. URL: https://github.com/DmitryKey/luke.