SEUPD2425-RACOON at LongEval: A Novel Approach To Information Retrieval With LLM-Based Query Expansion And Temporal Relevance Feedback Techniques

Notebook for the LongEval Lab at CLEF 2025

Giovanni Gaio[1], Filippo Mazzarotto[1], Matteo Meneghin[1], Enrico Saro[1], Francesco Visonà[1,*] and Nicola Ferro[1]

[1]University of Padua, Italy

Abstract

This paper presents the work of the Racoon group from the University of Padua on the LongEval *Conference and Labs of the Evaluation Forum (CLEF)* 2025 Lab in Task 1 "LongEval-Web Retrieval" [1]. The task was focused on developing *Information Retrieval (IR)* systems resilient to the temporal evolution of Web documents. To address this challenge, our approach combines traditional IR techniques with several novel components. Specifically, our *Search Engine (SE)* systems incorporate a French lemmatizer within the analyzer, a *Large Language Model (LLM)*-based *Query Expansion (QE)* module, a *Relevance Feedback (RF)* mechanism based on query assessment over the preceding months, and an LLM-based reranking strategy. Overall, our approach, which integrates Relevance Feedback with *Deep Learning (DL)* reranking, shows significant improvements over the baseline and maintains good short-term temporal robustness on the test set, although its effectiveness declines over the long term.

Keywords

Information Retrieval, Search Engine, Temporal Evolution, Relevance assessment, Query expansion, Reranking, Large Language Model, Conference and Labs of the Evaluation Forum

1. Introduction

A Search Engine is a software system designed to retrieve information from a collection of documents, based on user search queries. While this task has been addressed largely in literature through various approaches, maintaining robust effectiveness amid dynamic temporal changes in textual content remains an open challenge.

In this work, we set out to develop an IR system capable of sustaining its effectiveness over time. This was conducted within the framework of the first task "LongEval-Web Retrieval" of the "LongEval" Laboratory proposed by CLEF 2025[1]. The dataset consisted of a sequence of web document collections and queries, primarily in French, provided by Qwant[2], that spanned over nine months, capturing the evolution of documents and queries over time.

Our objective was to explore a number of different approaches, mostly standard techniques in various combinations, but also original ideas of our own. We started with a baseline Search Engine setup based on Lucene [2], and incrementally extended it by experimenting with different configurations for analyzing the documents and, more significantly, with a range of strategies aimed at improving their retrieval. As LLMs have gained widespread attention, praise, and adoption in recent times, we were particularly interested in evaluating their potential integration within our pipeline. We were ultimately limited by our computational resources in the depth of our tests, but we still obtained several noteworthy results.

CLEF 2025 Working Notes, 9 – 12 September 2025, Madrid, Spain

*Corresponding author.

giovanni.gaio.1@studenti.unipd.it (G. Gaio); filippo.mazzarotto@studenti.unipd.it (F. Mazzarotto); matteo.meneghin@studenti.unipd.it (M. Meneghin); enrico.saro@studenti.unipd.it (E. Saro); francesco.visona.3@studenti.unipd.it (F. Visonà); nicola.ferro@unipd.it (N. Ferro)

[1]For further information see: https://clef-longeval.github.io/tasks/.

[2]https://about.qwant.com/en/

The paper is organized as follows: Section 2 discusses the ideas we were inspired by; Section 3 describes our approach; Section 4 explains our experimental setup; Section 5 discusses our main findings and compares the various approaches; finally, Section 6 draws some conclusions and outlooks for future work.

2. Related work

We incorporated into our work insights from various approaches presented in prior studies that addressed the problem of longitudinal evaluation of IR systems, particularly within the CLEF 2024 LongEval Laboratory.

Specifically, the ideas of *Leveraging Prior Relevance Signals in Web Search* [3] were of particular interest to us, as they focused on the use of relevance assessments as a form of Query Expansion or reranking to obtain better effectiveness over time. Their work described how these methods could improve the mean nDCG of up to 8%.

We also focused on the methodology of Team Mouse in 2024 [4] which described how to use LLMs for Query Expansion and reranking through a reranker API. The latter approach was developed but not used due to constraints on the rate limits of such services.

The prompt design for the Query Expansion phase, discussed in Section 3.3.1, was primarily driven by a trial-and-error approach, drawing inspiration from the works *Query Expansion by prompting Large Language Models* [5] and *Corpus-Steered Query Expansion with Large Language Models* [6].

3. Methodology

This section describes the architecture of our system, which was developed primarily in Java using the *Apache Lucene* library, and in Python for some specific tasks of the Searcher component.

Our initial objective was to implement a fully functional pipeline, which was subsequently extended to improve effectiveness. The system is structured into four main components:

- **Parser**: a standard JSON parser designed for the document collection. It uses the *Jackson* library[3].
- **Analyzer**: responsible for text processing of the documents, including tokenization and the application of filters. It is described in Section 3.1.
- **Indexer**: constructs an index of the tokens obtained from the parsed and analyzed documents. It contains the document IDs and the terms of their bodies, other than a full text field that is used in the reranking phase, as described in Section 3.2.
- **Searcher**: manages query processing and document retrieval by adopting some strategies like Query Expansion, Relevance Feedback and reranking. See Section 3.3 for further details.

The following sections provide a more detailed description of the system, with a focus on the components that have been most substantially improved. A diagram of the overall configurable architecture is shown later in Figure 2.

3.1. Analyzer

The analyzer is the component which is responsible for analyzing the raw text of the documents and turning it into a set of tokens that can be used for indexing and searching.

During our tests we tried different combinations of filters and tokenizers. The final pipeline is shown in Figure 1 and described below.

[3]https://github.com/FasterXML/jackson

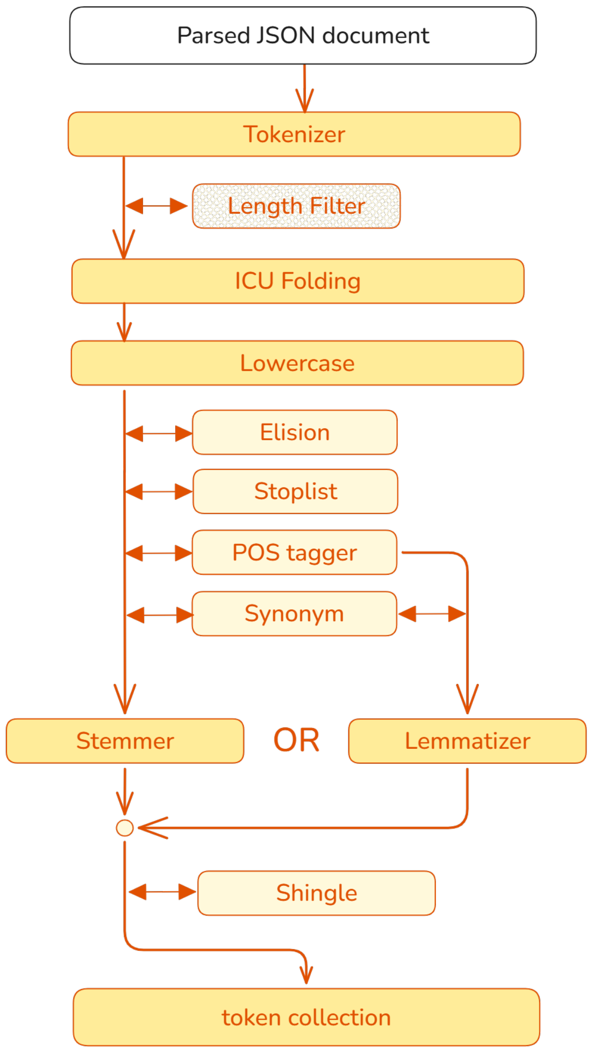


Figure 1: `RacoonAnalyzer` Pipeline.

3.1.1. Tokenizer

The first component of our pipeline is the tokenizer, which splits the text of each document into smaller units, producing a stream of tokens.

We tested three built-in tokenizers in *Apache Lucene*:

- `StandardTokenizer`: implements the Word Break rules from the Unicode Text Segmentation algorithm, as specified in Unicode Standard Annex 29 [7].
- `WhitespaceTokenizer`: divides text at any whitespace character.
- `LetterTokenizer`: divides text at non-letter characters.

3.1.2. Filters

The remaining pipeline was composed of various filters and the stemming step. The latter is described in more detail in Section 3.1.3. The rest of the filters are now described in the order they were applied:

1. *Length token filter (optional)*: filters out tokens that are too short or too long, given a minimum and maximum length.
2. *ICU folding*: applies search term folding to Unicode text, such as accent and diacritic removal.
3. *Lowercase*: converts all the text to lower case, which is better for information retrieval.

4. *Elision filter (optional)*: filters out elided tokens, such as *d'*.
5. *Stopword filter (optional)*: filters out stop words, such as articles, prepositions and conjunctions, based on an external list of words.
6. *POS tagging (optional)*: tags each token with its part of speech.
7. *Synonyms expansion (optional)*: expands each token with synonym, which are obtained from the WOLF database [8]. During its test, we moved the *ICU Folding* filter after this step.
8. *Shingle filter (optional)*: a Word N-gram filter that creates overlapping sequences of tokens and treats them as an index term.

3.1.3. Stemming and its alternatives

A key step of the pipeline is the normalization through stemming or lemmatization. Stemming is the process of reducing words to their root form, while lemmatization is the process of reducing words to their base dictionary form. We tested several built-in alternatives *Apache Lucene* provides, as well as a custom lemmatizer based on a large-scale French lexicon. The built-in solutions are:

1. **Light:** Applies a light stemming algorithm specifically designed for the French language, performing less aggressive suffix stripping than more intensive stemmers.
2. **Minimal:** Implements a minimal stemming algorithm for French, which typically removes only the most common inflections.
3. **Snowball:** Utilizes the Snowball stemming algorithm (often an evolution of the Porter stemmer) with rules tailored for the French language.
4. **NGram:** Applies a character-level N-gram token filter. This method breaks tokens into sequences of N characters. It is distinct from stemming as it does not aim to find the morphological root form of a word but instead represents tokens through their constituent character N-grams.

Our custom **Lexicon lemmatizer** is based on the *Lexique des formes fléchies du français (LEFFF)* [9], a large-scale morphological and syntactic lexicon for French. It contains roughly 600,000 entries, which map inflected forms to their lemmas, covering inflections for words (including multi-word entities) and punctuation.

Each LEFFF entry typically has four fields: the *inflection* (the word form as it appears in text), the *part of speech* (e.g., verb, noun), the *lemma* (the dictionary base form), and the inflected form's *grammatical features*. For example, an entry like '`persistait,v,persister,I3s`' maps the word "persistait" to its lemma "persister" when "persistait" is identified as a verb (v).

Our lemmatizer uses each lexicon entry as a rule: for a given token and its pre-assigned *Part-of-Speech (POS)* tag, if a matching inflected form is found in the LEFFF lexicon, the token is replaced with the corresponding lemma. During our testing, we moved the *ICU Folding* filter (for accent and diacritic removal) after the lemmatization step, obtaining great results on example sentences.

However, the tests used to assess this initial effectiveness were grammatically correct and contained correctly-placed diacritics. This is in contrast to real queries, which usually lack diacritics and contain errors or variations that do not perfectly match the strict entries in the LEFFF lexicon. Ultimately, this led to a poor effectiveness in practice.

3.2. Indexing

The indexer is a standard way to index documents in memory. The `IndexWriter` of *Apache Lucene* was used, storing the IDs (not filtered) and the tokens of each document. We also stored the term vectors and the positions of each token to be able to retrieve the documents from the index, which is really useful for some components of the searcher, like Relevance Feedback. Since most of the rerankers we tested required access to the full document text, we allocated a non-indexed field within our index to store a compressed version of each document's full text. While this decision doubled the overall index size, it significantly improved effectiveness during the reranking phase.

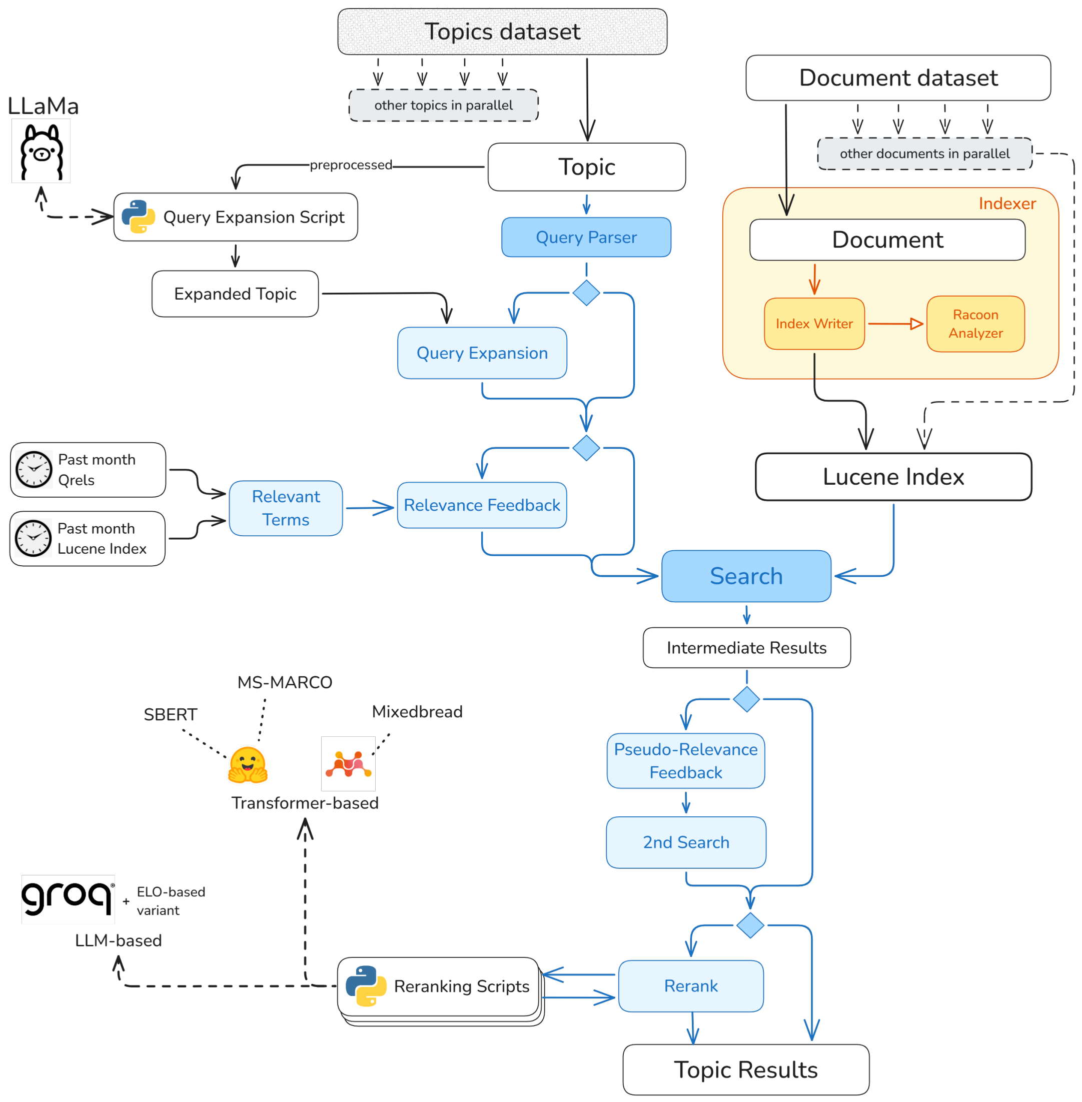


Figure 2: The architecture of our system, indexing documents in parallel using the `RacoonAnalyzer` and then searching all the topics. Diamonds represents conditional configurations.

We also considered the idea of storing the corresponding URLs of the documents, since many queries are incomplete URLs or refer to them, but decided against it given the poor results they had on the collection of the previous year (see for example [10]).

Given that this process is highly parallelizable, we used multiple threads to index the documents in parallel, so that the index step takes only about 2 minutes to finish for each month of the training set. In this phase it was critical to notice that many of the documents were duplicates, and subsequently store only one copy of them.

3.3. Searcher

The searcher is responsible for processing queries and retrieving documents from the index, making it a crucial component in IR systems. To maximize effectiveness, we developed a custom searcher that leverages `IndexSearcher` from *Apache Lucene*, executing each query in a separate thread.

In the standard search algorithm, we aimed to strike a balance between exact matches and highly similar results by employing the BM25 Similarity model [11]. Beyond this standard approach, implemented as a *Boolean Query*, we explored different methods to enhance effectiveness

1. Query Expansion, described in Section 3.3.1.

2. Relevance Feedback, described in Section 3.3.2.
3. Reranking, described in Section 3.3.3.

3.3.1. Query expansion

To enrich user queries with semantically related terms and improve information retrieval effectiveness, we implemented a Query Expansion pipeline using LLMs from Meta's LLaMA family, specifically LLaMA 3 70B [12] and LLaMA 4 Scout 17B [13], accessed through the Groq API[4]. The process aims to generate relevant query variations or semantic enrichments to boost recall in downstream retrieval components.

Prompt Design. The prompt design process was primarily driven by a trial-and-error approach, drawing inspiration from the works *Query Expansion by prompting Large Language Models* [5] and *Corpus-Steered Query Expansion with Large Language Models* [6].

Prompts were formulated in French to align with the language of the query logs. Ultimately, four distinct prompts were developed to explore different prompting strategies and to identify the best approach:

1. Generating 20 semantically related or contextually adjacent expressions to a query.

   "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."

2. Providing a passage that answers the query, along with a justification.

   "Rédigez un passage qui réponde à la question posée
   {query}
   Justifiez votre réponse avant de répondre. N'ajoutez pas de texte supplémentaire"

3. Listing synonyms and conceptually related terms.

   "Vous êtes un assistant de recherche d'informations. Etant donné la requête {query}, générez une liste de 10 termes sémantiquement liés, de synonymes et de mots-clés conceptuellement adjacents qui pourraient être utilisés pour étendre cette requête afin d'obtenir une meilleure couverture. Imprimez uniquement les termes sur une seule ligne, séparés par des virgules, n'ajoutez pas de texte ou d'explications supplémentaires."

4. Inferring the user's intent explicitly.

   "Vous trouverez ci-dessous une requête écrite par un utilisateur français à un moteur de recherche. Expliquez quelle est l'intention de la personne qui l'a écrite et répondez à la requête s'il s'agit d'une question. Commencez par 'L'utilisateur veut'. N'ajoutez pas de texte supplémentaire
   Requête: {query}"

All prompts instructed the LLM to avoid adding any additional text, in order to minimize noise in the output. In Prompt 4, the model was explicitly asked to begin its response with *"L'utilisateur veut"*, to prevent variations that might introduce undesired leading terms. These initial words were subsequently filtered out.

Models and APIs. We used the `llama3-70b-8192` model as the primary generator due to our constraints on rate limits, and `meta-llama/llama-4-scout-17b-16e-instruct` as a fallback for ambiguous or error-prone completions. Only when all these models failed, our system did not find any expansion term. Responses were streamed from the Groq API with temperature 1.0 and a maximum of 1024 tokens, which allowed us a restricted number of queries per day and per minute: that's the main reason for the implementation of the caching strategy.

[4]https://console.groq.com

Caching and Efficiency. Since the expansion process is time-consuming and subject to rate limit constraints, we implemented a caching strategy to prevent latency at search time: we asked the LLM to expand the queries and we stored the results in output files indexed by month and prompt type before running our system. This limitation would likely not be needed if the limits on the Groq API were not so strict or we had better hardware. These limits were respected using delays and token usage monitoring via response headers.

Weights of expanded terms. In this section, but also in the Relevance Feedback one (Section 3.3.2), we tried different ways to weight our expanded terms.

- **Constant weight:** the same weight is given to all the terms, which after some experiments was set to 0.5.
- **Position weight:** the weight of term i, where i refers to the position of the term in the expanded sequence, is calculated as:

$$w_i = \max\left(0.1, 0.5 \cdot \left(1 - \frac{i}{E}\right)\right)$$

  where E is the total number of expanded terms. This formula was primarily focused on the prompts asking for related terms, with the implicit assumption that the first ones were the most relevant.
- **IDF-like weight:** the weight of term i is calculated as:

$$w_i = 0.1 + \min\left(0.4, \frac{\log\left(N/(n_i+1)\right)}{10}\right)$$

  where n_i is the number of documents containing term i and N is the total number of documents. The formula was obtained by modifying the smoothed inverse document frequency dividing it by 10, while also ensuring a minimum boosting of 0.1 and a maximum of 0.5.

The best results for Query Expansion were obtained using a constant weight of 0.5.

3.3.2. Relevance feedback

The main focus of the task was to understand how our system could leverage the knowledge of previous months relevance to handle changes over time. Although the use of relevance assessments may seem unrealistic in practice, in a production environment they could be easily derived from implicit feedbacks like clicks, as it was done to build the CLEF LongEval training collection, or from explicit assessments of real users.

Our work, which was based on the ideas of *Leveraging Prior Relevance Signals in Web Search* [3], aims to use the information of relevance of prior months for a given query to boost relevant results for the same query in the current month. While this method is limited to topics with a known history, in this specific task the great overlapping between queries of different months allows the method to be very effective.

Related work. The focus of [3] is on two different approaches: on the one hand, it uses relevance assessments as a form of Query Expansion, adding terms of prior relevant documents to the query itself. The other approach leverages relevance assessments during the reranking phase to prioritize results that were more relevant in the previous months. This is achieved by assigning weights to documents based on their presence in previous evaluations, with newer relevant documents receiving higher weights than older ones.

Our work. We developed both the approaches of [3]: the first one is described below, while the reranking method can be found in Section 3.3.3.

The following Query Expansion methods were tested:

- *Pseudo RF (PRF)*: This is a standard PRF approach. We perform the standard query, and then we retrieve the 10 most frequent terms in the 5 most relevant documents and add them to the query itself.
- Relevance Feedback: This is a more complex approach, in which, before performing any retrieval, we first check whether the same query was submitted in previous months and whether we have relevant documents available for the query. If so, we retrieve the 8 most relevant documents for that topic and extract their most frequent 20 terms, which are then added to the current query. The assumption is that what was relevant in the previous months can help with the same query in the current month.
  In the following, and in Section 5, we refer to two versions of this mechanism. The first one, referred to as ONERF, considers only the relevance assessment from the previous month, i.e. the most recent relevance assessment available. The second approach, dubbed ALLRF, takes into account all the available relevance assessments of the previous months, and uses the most recent one for each query.
- Relevance Feedback and PRF: This combines the two previous approaches to integrate the strengths of both methods.

All the above numbers were obtained after a fine tuning phase on the training dataset.

As described in Section 3.3.1, we tried different weights to give to the new terms added to the query:

- **Constant weight**
- **Position weight**
- **IDF-like weight**

The best results for Relevance Feedback were obtained using a constant weight of 0.5.

3.3.3. Reranking

After retrieving relevant documents, initially scored using the Okapi BM25 ranking function, we selected the highest-scoring ones and updated their relevance using a more complex and computationally expensive procedure. As this is a standard step in modern IR systems, we could rely on a a rich set of existing work in this area. In some cases, the number of reranked documents was ultimately limited by the hardware resources available to us.

SBERT vector embedding cosine similarity score We modified the previously computed score for each retrieved document by adding the cosine similarity between the vector embeddings of the document and the query. These embeddings were calculated using a pretrained model available in the python library `sentence_transformers` [14, 15].

We tried different models from this library, but, given the limited time and computational power at our disposal, we selected `sentence-transformers/distiluse-base-multilingual-cased-v1` as a good "standard" model, as it is big and multilingual while also not being too slow, thus allowing us to re-rank 100 documents per query by simply adding the new similarity scores multiplied by 1.5.

Text Ranking models We also conducted various tests using two additional models from the same library: `cross-encoder/ms-marco-MiniLM-L6-v2`[5] and `mixedbread-ai/mxbai-rerank-base-v1`[6] [16].

[5]https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2
[6]https://huggingface.co/mixedbread-ai/mxbai-rerank-base-v1

These models are specifically aimed at reranking and directly provide a similarity score between the topic and document provided. While the first one (dubbed MARCO in the following) is smaller and faster, allowing to re-rank 100 documents, the second one (dubbed MIXEDBREAD) is bigger and slower, so we set it to re-rank only 60 documents. Also in this case we multiplied the similarity scores by 1.5.

LLM-based Elo pseudo ranking This approach is based on the assumption that LLMs, like humans, are better at comparing items than assigning scores. The idea is to sort the documents by comparing them two at a time (it is a sort of pairwise approach, as opposed to the previous pointwise approaches). The main difficulty is that doing a complete sorting requires a number of comparisons in the order of $\mathcal{O}(n \log n)$, which is too computationally expensive, at least given our limited resources.

For these reasons we tried to update the scores using an Elo ranking scheme . The Elo rating scheme is a method for calculating the relative strengths of entities, which was originally developed to rate chess players [17], but it is also used in contemporary research, for example to compare LLMs [18].

We selected some random documents and asked an LLM to rank them by relevance to the given topic.

We then update the score S for each document according to the following rule:

$$S_i^{\text{new}} = S_i^{\text{old}} + k(R - E)$$

Where E is the expected number of comparisons where the document i ranked better than the other; R is the actual number of comparisons where the document was better, for the ordering given by the LLM.

For each comparison we estimate the probability that a document d_A is better than document d_B, with respective scores S_A and S_B, is $\mathbb{P}(d_A > d_B) = \frac{1}{1+\exp(0.5\cdot(S_A - S_B))}$. Then to find R we simply sum over all documents: $E_A = \sum_{d \in D \setminus \{d_A\}} \mathbb{P}(d_A > d)$.

We were planning on using some freely available API to answer the prompt. This was ultimately not possible because the method requires a large number of queries and long prompts. We also tried using smaller models on the hardware we had. Also this approach was not feasible, thus we could not evaluate it.

Relevance feedback reranking. This approach, dubbed QRELS, is based on what has already been explained in Section 3.3.2. The idea is to use the query assessments from previous months to re-rank the retrieved documents: if a document was relevant in the past, we assign it a higher score. However, this strategy can easily lead to stale results, therefore we introduce a weight that favors more recent assessments, following the formula:

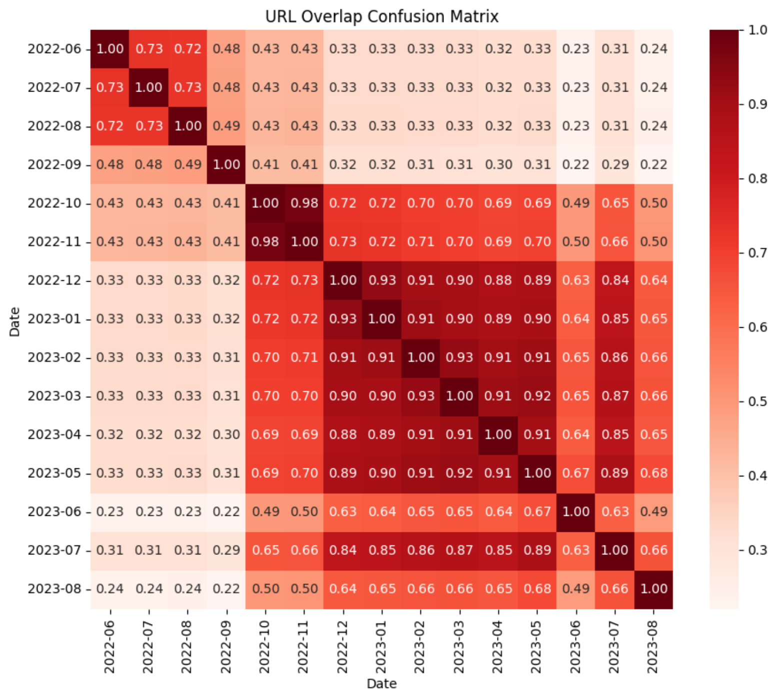
$$S_i^{new} = S_i^{old} + \alpha \cdot 0.8^{M_{diff}}$$

where M_{diff} refers to the difference between the current month and the month of the last relevance assessment, while α is a value that depends on the prior relevance of the document: 1.0 if the relevance was equal to 2, 0.5 if the relevance score was 1.

API reranking. This approach relies on different services that allow to re-rank automatically documents for a given query by simply passing the query and the top documents. While this method could be really useful, we could not use it due to the high volume of queries to process and the limited rate limits of the services. Nevertheless, it could be really interesting to try in the future.

4. Experimental setup

The source code of our information retrieval system is available online at the following address: https://bitbucket.org/upd-dei-stud-prj/seupd2425-racoon/src/master/.



Date	2022-06	2022-07	2022-08	2022-09	2022-10	2022-11	2022-12	2023-01	2023-02	2023-03	2023-04	2023-05	2023-06	2023-07	2023-08
2022-06	1.00	0.73	0.72	0.48	0.43	0.43	0.33	0.33	0.33	0.33	0.32	0.33	0.23	0.31	0.24
2022-07	0.73	1.00	0.73	0.48	0.43	0.43	0.33	0.33	0.33	0.33	0.32	0.33	0.23	0.31	0.24
2022-08	0.72	0.73	1.00	0.49	0.43	0.43	0.33	0.33	0.33	0.33	0.32	0.33	0.23	0.31	0.24
2022-09	0.48	0.48	0.49	1.00	0.41	0.41	0.32	0.32	0.31	0.31	0.30	0.31	0.22	0.29	0.22
2022-10	0.43	0.43	0.43	0.41	1.00	0.98	0.72	0.72	0.70	0.70	0.69	0.69	0.49	0.65	0.50
2022-11	0.43	0.43	0.43	0.41	0.98	1.00	0.73	0.72	0.71	0.70	0.69	0.70	0.50	0.66	0.50
2022-12	0.33	0.33	0.33	0.32	0.72	0.73	1.00	0.93	0.91	0.90	0.88	0.89	0.63	0.84	0.64
2023-01	0.33	0.33	0.33	0.32	0.72	0.72	0.93	1.00	0.91	0.90	0.89	0.90	0.64	0.85	0.65
2023-02	0.33	0.33	0.33	0.31	0.70	0.71	0.91	0.91	1.00	0.93	0.91	0.91	0.65	0.86	0.66
2023-03	0.33	0.33	0.33	0.31	0.70	0.70	0.90	0.90	0.93	1.00	0.91	0.92	0.65	0.87	0.66
2023-04	0.32	0.32	0.32	0.30	0.69	0.69	0.88	0.89	0.91	0.91	1.00	0.91	0.64	0.85	0.65
2023-05	0.33	0.33	0.33	0.31	0.69	0.70	0.89	0.90	0.91	0.92	0.91	1.00	0.67	0.89	0.68
2023-06	0.23	0.23	0.23	0.22	0.49	0.50	0.63	0.64	0.65	0.65	0.64	0.67	1.00	0.63	0.49
2023-07	0.31	0.31	0.31	0.29	0.65	0.66	0.84	0.85	0.86	0.87	0.85	0.89	0.63	1.00	0.66
2023-08	0.24	0.24	0.24	0.22	0.50	0.50	0.64	0.65	0.66	0.66	0.65	0.68	0.49	0.66	1.00

Figure 3: Overlap matrix of the documents in the training and test collection. Available at https://clef-longeval.github.io/data/

Hardware setup The Search Engine and all suitable tests were run on the following system:

- *Operating System (OS)*: Windows 11 Home,
- 32 GB of RAM,
- AMD Ryzen 9 6900HX with Radeon Graphics. The absence of a dedicated NVIDIA GPU is a limitation, which was overcome by running some parts of the reranking phase on a separate computer.

To allow easy and fast use of LLMs, we also took advantage of the online service groq.com, this was done mostly for Query Expansion. We mostly used Llama 3 70B as LLM, while occasionally asking Llama 4 17B Scout to expand the most complex queries.

In some cases, to accelerate inference with machine learning models, we ran just this part on another computer. This second computer communicated via *HyperText Transfer Protocol (HTTP)* with the master computer described above and was equipped with an Nvidia GeForce GTX 960 with 4GB VRAM, 16 GB RAM, an Intel Core i5 6500 CPU, and running the OS Manjaro Linux.

Training data The training data was provided by the organizing committee of CLEF 2025 [19], and is publicly available together with test data on the TU Wien Research Data Repository [20]. It consists of 9 monthly data snapshots, acquired from June 2022 to February 2023 and composed of documents (18M), queries (9k) and qrels (relevance assessments). Both the documents and the queries vary in the 9 snapshots, but there is a high correlation between different months, both in the documents, as shown in Figure 3, and in the queries, as shown in Figure 4. The queries were extracted from Qwant's search logs and based on a set of selected topics, while the documents are all the web pages that have been displayed in *Search Engine Result Pages (SERPs)* for the selected queries. Filters were applied to exclude adult content from the collection. The relevance estimates were obtained through automatic collection of user implicit feedback and are on a 3-level scale. For more information please refer to https://clef-longeval.github.io/data/.

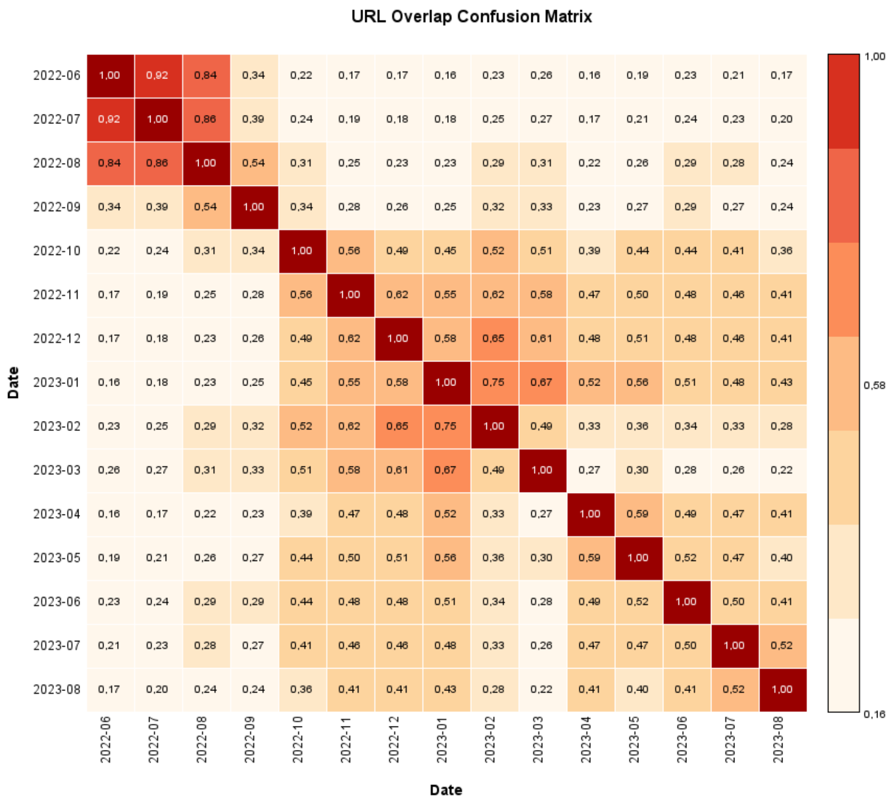


Figure 4: Overlap matrix of the queries in the training and test collection.

Test data Test data was composed of six months of queries and documents, from March 2023 to August 2023. As can be seen in Figure 3, there is a high overlap of documents with the training data, although the queries share fewer similarities (see Figure 4). In particular, it is instructive to notice that the overlap between the queries of the months of the test data and the last month of the training data (2023-02) is fairly low, which will cause our ONERF mechanism to fail. The relevance assessments were obtained in the same way as for the training data, but they are not available.

Evaluation measures The submitted systems will be evaluated using the *Normalized Discounted Cumulated Gain (nDCG)* and *Relative nDCG Drop (RnD)* between different snapshots of the test set, as these are the evaluation metrics used in the CLEF conference[7]. For a detailed description of the metrics, see [21].

The nDCG is a well-known evaluation measure calculated on a set of queries Q.

The RnD is a normalized version of the difference of nDCG between snapshots, which has the following formula:

$$\mathrm{RnD}(old, new) = \frac{\mathrm{nDCG}(old) - \mathrm{nDCG}(new)}{\mathrm{nDCG}(old)}$$

where new is the newer snapshot and old the older. This measure supports the evaluation of the impact of the data changes on the systems' results: small absolute values indicate a stable system, positive values indicate a degradation, and negative values indicate an improvement.

Given that our systems will be evaluated on nDCG, all the experiments of Section 5 will focus on the nDCG measure, and make decision based on it.

[7]https://clef-longeval.github.io/tasks/

Statistical analysis For both the training and test data we first conducted a 2-way ANOVA on topics and systems to determine if some of them were statistically different from the others, and then a Tukey HSD test to determine which systems were different. The confidence level was set to 0.05.

We report graphs of the results of Tukey HSD tests, while we do not report ANOVA results for the sake of space.

5. Results and discussion

The results are divided in two parts: in Section 5.1 we describe how we obtained the systems that we submitted to the CLEF 2025 evaluation, while in Section 5.2 we discuss the results of the CLEF 2025 evaluation of our systems. For a complete comparison with other systems presented at the conference, see [22].

5.1. Training data

As we proposed and tried many different possibilities for the different parts of our Search Engine, we could not test all possible combinations. Therefore we proceeded in the following order:

1. We first selected the best analyzer, retrieving documents with a fixed base searcher;
2. In the second phase, we compared the results of ONERF and PRF using the best analyzer obtained previously;
3. After assessing the really good impact of ONERF, we compared the rerankers, using the best analyzer found previously and ONERF;
4. In the fourth phase we assessed the need for Query Expansion, choosing the best prompts;
5. Finally, we compared the best combination of ONERF, reranker and Query Expansion to decide the systems that we would submit.

We omit for brevity the tests that were necessary to fine tune some of the parameters of both the analyzer and the searcher, like the weights for the different phases. This fine-tuning was done on different months to avoid overfit to a specific document collection. Also, some tests that could have been performed (like testing if using URLs could lead to better results) were not done since they had not yielded good outcomes in previous years. In the following paragraphs we will present a summary of the results of the previously described phases.

Phase 1: Analyzer To select the analyzer we tried different combinations of the filters (for reference, see Section 3.1) while keeping the searcher fixed.

The results clearly showed that the worst systems were the ones with the character N-gram filter (as expected, since in general it performs worse than any other stemmer) and the one with our custom Lefff stemmer, which could be explained by the fact that the queries didn't have any diacritics, which reduced the effectiveness of the overall system.

It was also interesting to see that adding synonyms using the WOLF database gave bad results: that was probably due to the quality of the database, but also to the fact that adding synonyms can in general lead to topic drift. What's more, using other filters like French elision and word N-gram seemed to deteriorate effectiveness.

In the end we concluded that the best possible analyzer was composed of the following phases:

- LetterTokenizer;
- ICU Folding filter;
- Lowercase filter;
- Stopwords filter, which used one of the best-performing stoplists from previous years' evaluations, plus the 30 most frequent words in the corpus;
- Light stemmer.

Phase 2: Relevance feedback and pseudo relevance feedback Phase 2 regarded the main task of the project: using prior knowledge to leverage the system's effectiveness on new months.

In this phase we tested two different approaches, alone and combined (see Section 3.3.2 for more details):

- PRF;
- ONERF: this method searches for relevant documents only on the immediately previous month, if there is a relevance assessment available. In the case of the test data, since data from the previous month is not available, the relevance assessments of the last training month are used.

The results showed that using prior knowledge of relevance on the same queries could significantly improve system's effectiveness. Moreover, this did not lead to stale results: we were not boosting older documents directly, but using their terms to obtain better retrieval quality. A major issue for the base system was the number of queries for which it returned zero results. With the ONERF mechanism, this number was reduced by up to a factor of 2 across all months.

The PRF mechanism, on the other hand, performed worse than expected, even if used in combination with ONERF. This was probably due to its greater tendency to cause a complete topic drift, as we are not certain that the top-ranked documents are actually relevant.

Phase 3: Reranking The third phase we tested a number of the rerankers that were developed. It is worth noticing that we tested only SBERT, MARCO, MIXEDBREAD, and the one based on previous relevance assessments, dubbed QRELS. Due to resource constraints, we were unable to evaluate the reranker based on the ELO method or those requiring the use of an API. We also investigated whether using the full text documents as input would lead to significant improvements.

The results showed that, in general, systems with a reranker based on full text performed better. This is likely caused by the fact that terms in the full text were not stemmed and therefore easier to use (by LLMs but also by other Deep Learning algorithms). Furthermore, using the appropriate reranker increased drastically the nDCG of the overall system.

The most disappointing reranker was QRELS, the one based on previous relevance assessments, whose effectiveness was nearly equivalent to the base system without reranking. This may be due to several factors, such as the possibility that boosting older documents may lead to stale results, especially with collections that are continuously updated.

In conclusion, the best reranker we obtained was MARCO, which was significantly better than the others, other than faster.

Phase 4: Query expansion The fourth phase focused on selecting the most effective Query Expansion prompt: as described in Section 3.3.1 we had 4 prompts to test, and therefore 4 systems.

Due to the high latency of the Query Expansion mechanism, we expanded only the first 200 queries from December 2022 using all four prompts, in order to assess which performed best. No reranker was used in this phase.

The results showed that, even if prompt 4 was the most performing, there were no statistically significant differences with the other ones, as they all performed worse than the base system. This outcome may be attributed to the need for further refinement of the prompts or the use of more advanced LLM models.

In the final phase we expanded all queries using prompt 1 and 4, as they represent the two most promising approaches to Query Expansion: asking for related keywords and asking to formulate explicitly the information need of the user.

Phase 5: Final comparison This comparison was conducted after completing all the previous phases, using the best configuration identified for each step. In the fifth and final phase, full-text reranking was consistently applied.

Base refers to a system that uses the best analyzer found in **Phase 1** and the fixed searcher that do not exploit any form of Query Expansion, Relevance Feedback, or reranking. We evaluated 12 systems:

- **Base+ONERF**: base system with ONERF;
- **Base+QE4**: base system with Query Expansion adopting prompt 4;
- **Base+QE1**: base system with Query Expansion adopting prompt 1;
- **Base+ONERF+QE4**: base system with ONERF and Query Expansion adopting prompt 4;
- **Base+ONERF+QE1**: base system with ONERF and Query Expansion adopting prompt 1;
- **Base+MARCO+ONERF+QE4**: base system with ONERF, MARCO reranker and Query Expansion adopting prompt 4;
- **Base+MARCO+ONERF+QE1**: base system with ONERF, MARCO reranker and Query Expansion adopting prompt 1;
- **Base+MARCO**: base system with MARCO reranker;
- **Base+MARCO+ONERF**: base system with MARCO reranker and ONERF;
- **Base+MBREAD+ONERF**: base system with MIXEDBREAD reranker and ONERF;
- **Base+SBERT+ONERF**: base system with SBERT reranker and ONERF;
- **Base+QRELS+ONERF**: base system with reranker based on previous relevance assessments and ONERF mechanism.

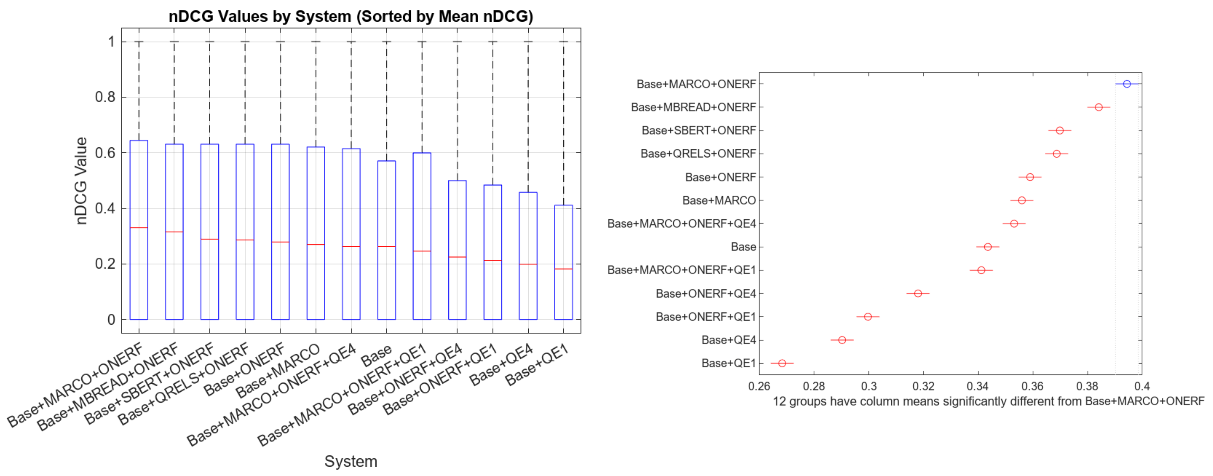


Figure 5: On the left, boxplot of nDCG of the 12 systems on October 2022 dataset, ordered by nDCG. On the right, the result of the Tukey HSD test on the 12 systems.

While the ANOVA test confirmed that the systems were statistically different, Figure 5 clearly illustrates the benefits of including a reranking phase on top of the ONERF mechanism, as the top four systems all used relevance feedback and a reranker.

Other performing systems are those using the ONERF mechanism or MARCO alone, both of which outperformed all forms of Query Expansion. This is probably due to the fact that the expanded terms are highly likely to cause a query drift. Additionally, we note that prompt 4, which asks the LLM to state the user's information need, performed statistically better than prompt 1, which requests related terms.

5.2. Test data

Throughout this section, we refer to the **Base** system as introduced in **Phase 5** of Section 5.1: the system includes the best analyzer identified in **Phase 1** (composed, in order, of LetterTokenizer, ICU Folding filter, Lowercase filter, Stopwords filter, and Light stemmer) and a fixed searcher that does not leverage any form of Query Expansion, Relevance Feedback, or reranking.

After the previous statistical analysis, we selected the following systems for submission, taken from the best configurations found in **Phase 5** of Section 5.1:

- **Base+MARCO+ONERF**, dubbed **run1**: base system with MARCO reranker and ONERF;

- **Base+ONERF**, dubbed **run2**: base system with ONERF;
- **Base+MARCO+ONERF+QE4**, dubbed **run3**: base system with ONERF, MARCO reranker and Query Expansion adopting prompt 4;
- **Base**, dubbed **run4**: base system.

In the following, however, we will consider the comparison of their results with those of 4 other systems, not submitted to the CLEF 2025 evaluation:

- **Base+PRF**, dubbed **run5**: base system with PRF;
- **Base+QRELS+ONERF**, dubbed **run6**: base system with QRELS reranker and ONERF;
- **Base+ALLRF**, dubbed **run7**: base system with a Relevance Feedback mechanism that searched relevant assessments for the query not only in the previous month, but also in the all the other ones, dubbed ALLRF. The difference between the standard ONERF mechanism used in **run1**, **run2**, **run3** and **run6** and the one used in this system, also explained in Section 3.3.2, was not so relevant in the training phase, but becomes more important in test data, since most of the queries of the test months do not overlap with the ones of February 2023, which is the last relevance assessment available, as depicted in Figure 4;
- **Base+MARCO+ALLRF**, dubbed **run8**: base system with MARCO reranker and a Relevance Feedback mechanism analogous to **run7** (ALLRF);
- **Base+MARCO**, dubbed **run9**: base system with MARCO reranker.

As for **run7** and **run8**, they were not included among the systems submitted to the CLEF 2025 evaluation, as the limitations of the ONERF mechanism (outlined in Section 5.1) only became apparent after analyzing the results on the test data. From this observation the modifications to obtain the ALLRF system was simple.

In the following sections, we analyze the effectiveness of our systems. It is important to note that a complete evaluation is difficult, due to the lack of results from systems developed by other teams. Therefore, our evaluation is based on the difference in the metrics (nDCG, RnD) relative to the base system, **run4**.

Moreover, we note that our systems were evaluated using the relevance assessments provided by CLEF, excluding any queries for which no relevant documents were identified among those assessed. As a result, our evaluation metrics may slightly differ from those reported by other teams.

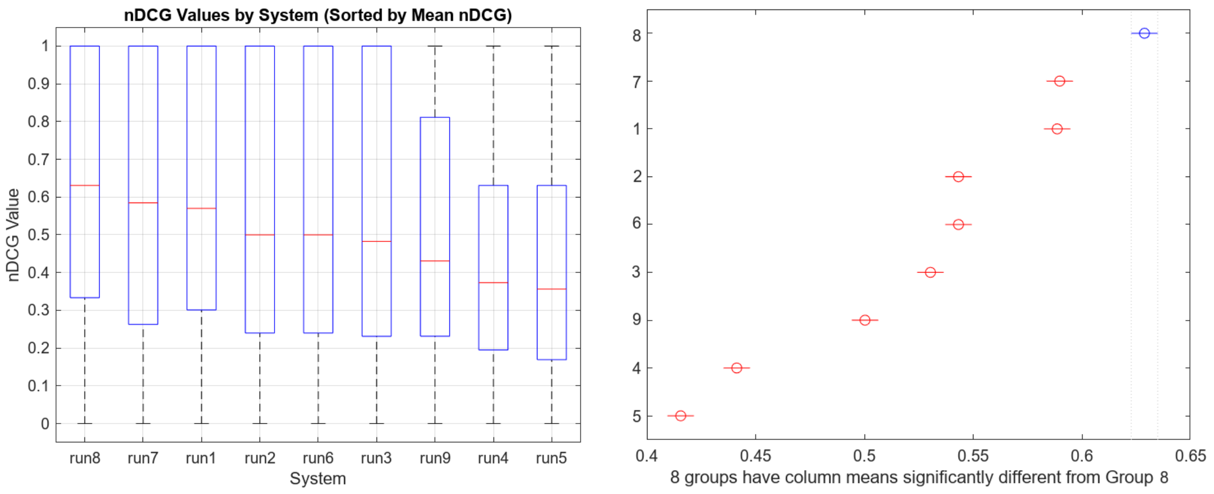


Figure 6: On the left, boxplot of nDCG of the 9 systems on March 2023 dataset, ordered by nDCG. On the right, the result of the Tukey HSD test on the 9 systems (the labels on the y axis indicate the number of the run).

Results and discussion on March dataset. Figure 6 reports the results on the March 2023 dataset, which is the closest to the most recent relevance assessments available from the training set. Figure 6 clearly shows the benefit of using any form of Relevance Feedback. Specifically, the results of **run2** (**Base+ONERF**) and **run1** (**Base+MARCO+ONERF**) are significantly better than those of **run4** (**Base**) and **run9** (**Base+MARCO**).

More importantly, we note that the use of the ALLRF mechanism considerably improves the effectiveness of all systems, with **run7** (**Base+ALLRF**) and **run8** (**Base+MARCO+ALLRF**) achieving higher scores than all others.

As already observed in Section 5.1, the Query Expansion mechanism does not appear to be effective in practice. Although **run3** returns results for nearly all queries, its overall effectiveness is substantially lower than that of **run1**. Similarly, the PRF mechanism (**run5**) degrades the overall system's effectiveness.

Our custom reranker based on previous relevance assessments, i.e. **run6**, performs better than systems without any reranker, but it is not as effective as Deep Learning based rerankers like MARCO (see for example **run1**).

Overall, the best results are obtained with **run8**, which combines the ALLRF mechanism with the MARCO reranker. Nevertheless, the ONERF mechanism based solely on the previous month also yields competitive results in this first month.

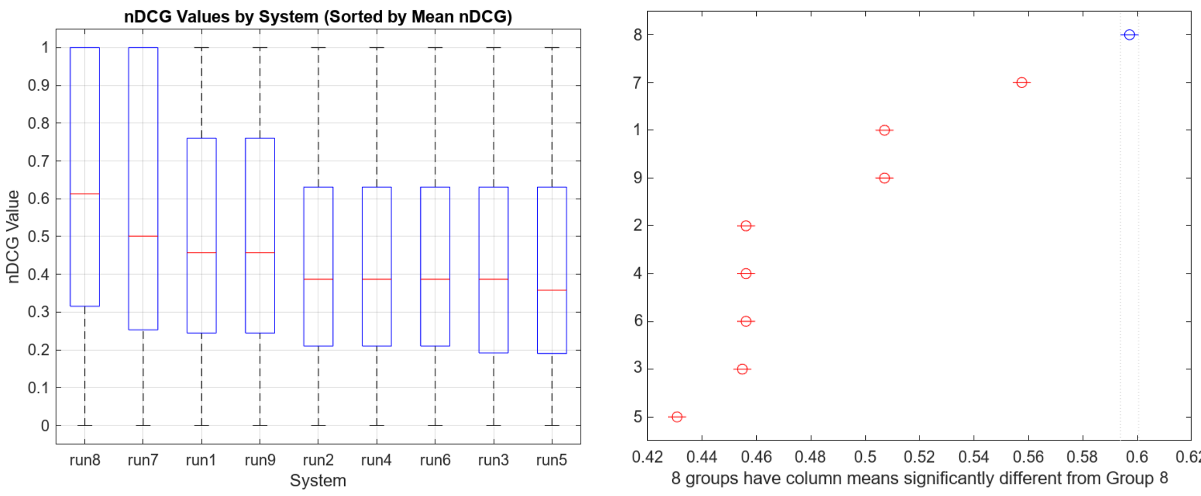


Figure 7: On the left, boxplot of nDCG of the 9 systems on April 2023 dataset, ordered by nDCG. On the right, the result of the Tukey HSD test on the 9 systems (on the y axis there is the number of the run).

Results and discussion on April dataset. The results change in the following months, such as April 2023. As shown in Figure 7, the ONERF mechanism based only on the previous month and the QRELS reranker perform comparably to the base system. In this case, nearly all the improvement in **run1** is attributable to the MARCO reranker, given that its effectiveness is not statistically different from **run9**.

On the other hand, the ALLRF mechanism and the Deep Learning based rerankers are the only techniques that provide a statistically relevant improvement over the base system, and appear to maintain their effectiveness over time. This is most likely caused by the higher probability of ALLRF to find relevance assessments for the queries, since as shown in Figure 4 this overlap is higher with months previous to February 2023.

Results and discussion on test data. Table 1 presents the average effectiveness of our systems across the entire test dataset. The results are broadly consistent with those observed for exclusively the April dataset, and similar conclusions can be drawn. The best-performing configuration is **run8** (**Base+MARCO+ALLRF**), achieving an average nDCG of 0.583.

Table 1
Average nDCG across the months of the test data, between March 2023 and August 2023, for the 9 systems.

System	1	2	3	4	5	6	7	8	9
Average nDCG on test data	0.505	0.455	0.453	0.438	0.413	0.455	0.544	**0.583**	0.490

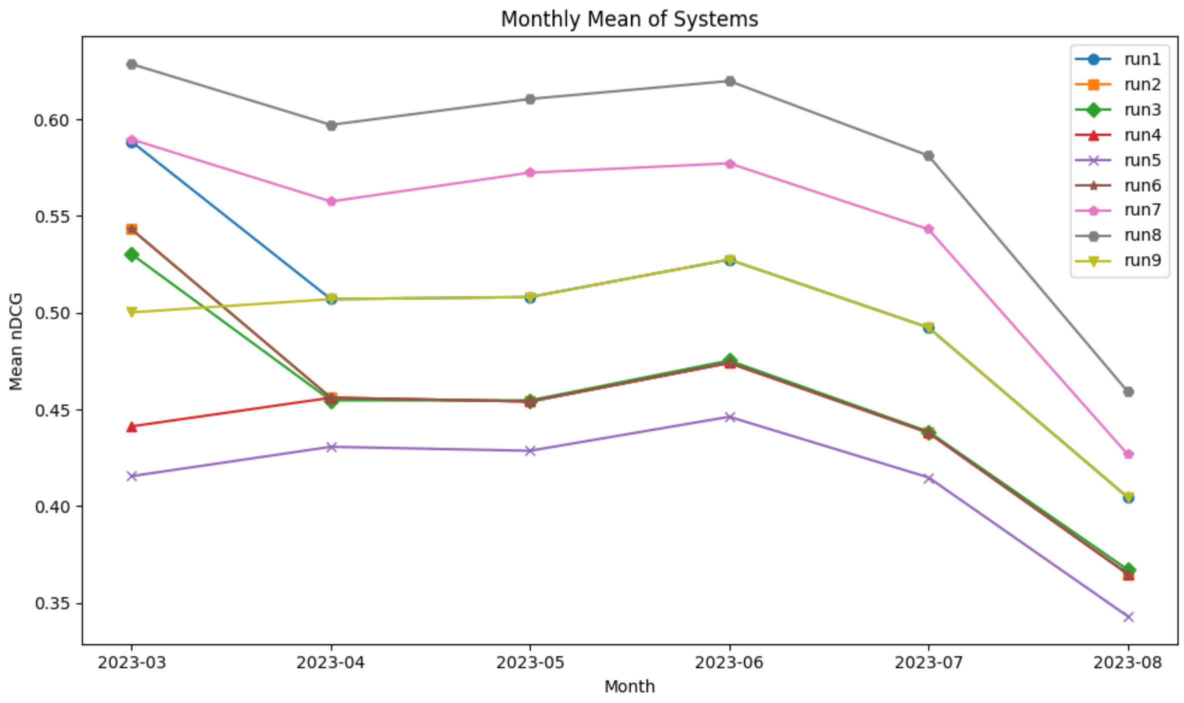


Figure 8: Mean nDCG of the 9 systems over all the monthly datasets.

Overall, systems incorporating the ALLRF mechanism (**run7**, **run8**) consistently outperform those using ONERF (**run1**, **run2**), with an average nDCG improvement exceeding 15%. Similarly, the inclusion of the MARCO reranker leads to substantial gains: systems that incorporate MARCO (e.g., **run1** vs. **run2**, **run9** vs. **run4**) show an average effectiveness boost of approximately 11%.

In contrast, alternative strategies such as PRF, the QRELS-based reranker, and LLM-based QE result in diminished effectiveness. These methods yield lower average nDCG scores compared to systems that omit them, indicating that they likely introduce noise or topic drift.

Results and discussion on robustness of systems over time. Figure 8 illustrates the effectiveness evolution of the nine systems throughout the test period. The plot shows that all systems experience effectiveness decline, particularly pronounced during the final month. Moreover, it indicates that the ALLRF-based systems (**run7**, **run8**) maintain consistent effectiveness in the short term, while degrading in the long term. In contrast, the effectiveness of the standard ONERF approach (**run2**) declines after the initial month, converging to the effectiveness level of the base system (**run4**). The MARCO reranker (**run9**) also provides a stable improvement over the base system (**run4**), and it demonstrates a decline in effectiveness over time comparable to that of the base system.

These observations confirm that the ALLRF mechanism, identified earlier as the most impactful component in terms of average effectiveness gain, is also effective in maintaining robustness over the short term, but may lose its effectiveness in the long term.

Table 2
Relative nDCG drop between March 2023 and April 2023, between July 2023 and August 2023, between March 2023 and August 2023.

Relative nDCG drop (RnD)	1	2	3	4	5	6	7	8	9
$\frac{\text{nDCG(03)} - \text{nDCG(04)}}{\text{nDCG(03)}}$	0.138	0.160	0.142	-0.034	**-0.036**	0.160	0.054	0.050	-0.014
$\frac{\text{nDCG(07)} - \text{nDCG(08)}}{\text{nDCG(07)}}$	0.178	0.167	**0.163**	0.167	0.173	0.167	0.214	0.210	0.178
$\frac{\text{nDCG(03)} - \text{nDCG(08)}}{\text{nDCG(03)}}$	0.312	0.328	0.307	**0.173**	0.174	0.328	0.276	0.269	0.191

Results and discussion on relative nDCG drop. Table 2 reports the relative nDCG drop (RnD) observed between two consecutive months (March 2023 to April 2023 and July 2023 to August 2023), as well as between the first and the last month of the test period (March 2023 to August 2023). Our discussion will focus on the comparison between our systems and the base system (**run4**).

Some systems demonstrate strong temporal stability. In particular, **run5** (**Base + PRF**) and **run9** (**Base + MARCO**) exhibit very low RnD values, even negative in the short term, and only minimal deviation from the base system **run4**. This result is somewhat expected, as these systems do not rely on relevance assessments from previous months, making them less prone to effectiveness degradation due to temporal drift.

In contrast, systems based on relevance feedback, namely **run1**, **run2**, and **run6**, show higher degradation, both in the short term (RnD ≈ 0.16) and in the long term, with relative drops ranging from 0.31 to 0.33. Systems incorporating the ALLRF strategy (**run7**, **run8**) exhibit comparable, though slightly lower, long-term degradation (RnD ≈ 0.27), while achieving significantly better short-term robustness (RnD ≈ 0.05). These results indicate that leveraging relevance assessments from all previous months enhances temporal robustness beyond that of ONERF-based systems.

An inspection of Figure 8 reveals that all systems experience an effectiveness drop in the final month of the test set (August 2023). This drop is particularly pronounced for the ALLRF systems, whereas ONERF-based systems show a steeper degradation at the beginning of the test period, after which their effectiveness and decline aligns with that of the base system.

This observation motivates the inclusion in Table 2 of the RnD values for the last two months. As shown, all systems suffer a comparable drop in retrieval effectiveness during this interval (RnD ≈ 0.17), except for the ALLRF-based systems, which show a greater decline, RnD around 0.21.

A more comprehensive evaluation, based on repeated queries across multiple months, could provide deeper insights into this behavior. Furthermore, it could help determine the maximum temporal window within which past relevance assessments remain effective before their contribution starts to deteriorate effectiveness.

6. Conclusions and future work

6.1. Our achievements

In this work, we proposed and evaluated several techniques aimed at improving both the retrieval quality and the temporal robustness of an IR system.

Our custom LEFFF-based lemmatizer, which utilizes a French morphological lexicon, was found to be one of the least effective methods in practice. This outcome was in contrast to more promising initial assessments, but which relied on clean, well-formed sentences. This poor effectiveness can be primarily attributed to its sensitivity to the high frequency of misspellings and missing diacritics in user queries, which often failed to match the lexicon's strict entries and led to frequent lookup failures.

Another line of investigation focused on LLM-based Query Expansion. We utilized `llama3-70b-8192` and `meta-llama/llama-4-scout-17b-16e-instruct`, tested four different prompt templates, and implemented a caching mechanism to precompute expanded queries and mitigate API rate limits. Despite its theoretical appeal, this approach consistently resulted in reduced retrieval effectiveness in our experiments. The decline in effectiveness may be attributed to insufficient control over semantic drift, the difficulty in weighting expansion terms effectively, or suboptimal prompt engineering. Nevertheless, we believe this direction remains promising and deserves further exploration.

Among the different technique explored, the most successful was a Relevance Feedback mechanism based on relevance assessments of the same queries in previous versions of the document collection (dubbed ALLRF), which consistently improved effectiveness across all test datasets. When combined with a Deep Learning based reranker (MARCO), this method achieved the best overall results in terms of nDCG, with an average score of 0.583 and a 33% improvement over the base system.

We also investigated ONERF, a Relevance Feedback strategy that relied only on relevance assessments from the immediately preceding month (in the test set, this meant using relevance from the last training month). While the former outperformed the base system, replacing it with ALLRF led to an average effectiveness gain exceeding 15%. Unfortunately, this improvement was not evident with the training data, which led us to submit only ONERF-based systems to the CLEF 2025 evaluation. In contrast, classical PRF consistently degraded system effectiveness, performing worse than the base system. This empirically confirms the risk of topic drift inherent to PRF techniques.

We experimented with various reranking techniques. The most effective Deep Learning based reranker on the training data was MARCO, and it was therefore adopted in the final test phase. Other rerankers such as MIXEDBREAD and SBERT also showed good effectiveness, with only marginal differences compared to MARCO. Moreover, we implemented a reranking strategy based on prior relevance judgments (QRELS). While it performed comparably to Deep Learning based rerankers on training data, its effectiveness dropped significantly on test data, yielding results not statistically different from the baseline. Additionally, we conceptualized an Elo-based reranking approach using LLMs, but it was not feasible to implement it due to computational limitations.

With respect to temporal robustness, all systems exhibited a decline in effectiveness over time. As expected, the most stable systems were those that did not rely on historical relevance assessments, thereby avoiding the risk of temporal degradation. Test data analysis suggests that while ALLRF significantly improves overall nDCG compared to ONERF, it also guarantees better robustness on the short term. However, both approaches exhibit similar effectiveness degradation in the long term, indicating that the benefits of incorporating more extensive relevance history comes at the cost of increased temporal instability.

Overall, our findings demonstrate that combining Relevance Feedback mechanisms leveraging multiple temporal snapshots with Deep Learning based reranking constitutes an effective solution for IR, but it may degrades its effectiveness in temporally dynamic settings.

While other techniques showed less immediate success, they offer promising directions for future improvements.

6.2. Further development

Many of the ideas we proposed and tried allow further refinement, experimentation and extension. In the following we briefly outline several directions for improvement across key system components.

LEFFF-Lemmatizer The current lemmatizer is likely too strict and does not allow for enough flexibility in the matching process, in particular when dealing with accented forms. In the future, this limitation may be mitigated by preprocessing the LEFFF lexicon removing diacritics prior to lookup, and incorporating fuzzy matching techniques. Furthermore, a standard stemmer may be used when the lemmatizer fails to find a match, allowing for a fallback mechanism.

Reranking The QRELS reranking technique we tried was mostly a proof of concept: it demonstrates the potential value of leveraging older relevance assessments. In future work, implementing a more sophisticated freshness-aware weighting function could mitigate the risk of promoting outdated content.

In the reranking phase, limited computational resources constrained the number of tested Deep Learning models and the extent of hyperparameter tuning. With better hardware, we could explore fine-tuning existing models, experimenting with prompt variations, or leveraging more powerful multilingual transformers to improve accuracy and robustness.

Query expansion The same considerations apply to the QE phase, where our reliance on external LLM APIs imposed latency and rate-limit constraints. Future work should focus on optimizing prompt design, potentially through few-shot prompting, and developing more reliable heuristics for integrating expanded terms into the query.

Relevance Feedback We highlighted the importance of relevance feedback, particularly with the ALLRF mechanism, which searches for relevant documents across all prior relevance assessments. An interesting direction for further investigation is to determine how long these past assessments remain effective. Essentially, this means identifying at which point the mechanism begins to degrade in effectiveness or introduce topic drift. A preliminary indication is provided in Section 5.2 where we can see that systems' effectiveness drops significantly after six months. However, a more systematic analysis using repeated queries over longer periods could provide clearer insights.

Declaration on Generative AI

During the preparation of this work, the authors used Gemini, Mistral in order to: Paraphrase and reword, Grammar and spelling check. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] CLEF Organizers, Longeval lab - clef 2025, 2025. URL: https://clef-longeval.github.io/, accessed: 2025-05-02.

[2] A. Sharma, Practical Apache Lucene 8. Uncover the Search Capabilities of Your Application, Apress Media, New York, USA, 2020.

[3] J. Keller, T. Breuer, P. Schaer, Leveraging prior relevance signals in web search, in: G. Faggioli, N. Ferro, P. Galuscáková, A. G. S. de Herrera (Eds.), Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), Grenoble, France, 9-12 September, 2024, volume 3740 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2024, pp. 2396–2406. URL: https://ceur-ws.org/Vol-3740/paper-220.pdf.

[4] L. Cazzador, F. Faveri, F. Franceschini, L. Pamio, S. Piron, N. Ferro, Seupd@ clef: team mouse on enhancing search engines effectiveness with large language models, Faggioli et al.[12] (2024).

[5] R. Jagerman, H. Zhuang, Z. Qin, X. Wang, M. Bendersky, Query expansion by prompting large language models, 2023. URL: https://arxiv.org/abs/2305.03653. `arXiv:2305.03653`.

[6] Y. Lei, Y. Cao, T. Zhou, T. Shen, A. Yates, Corpus-steered query expansion with large language models, in: Y. Graham, M. Purver (Eds.), Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, St. Julian's, Malta, 2024, pp. 393–401. URL: https://aclanthology.org/2024.eacl-short.34/.

[7] Unicode Consortium, Unicode Standard Annex #29: Unicode Text Segmentation, Technical Report UAX #29, Revision 45, Unicode Consortium, 2024. URL: https://unicode.org/reports/tr29/, latest version available at https://unicode.org/reports/tr29/. This version corresponds to Unicode 16.0.0. Editor: Josh Hadley.

[8] B. Sagot, D. Fišer, Building a free French wordnet from multilingual resources, in: Proceedings of the OntoLex 2008 Workshop, Marrakech, Morocco, 2008.

[9] B. Sagot, The lefff, a freely available and large-coverage morphological and syntactic lexicon for french, 7th international conference on Language Resources and Evaluation (LREC 2010) (2010).

[10] F. Galli, M. Rigobello, M. Schibuola, R. Zuech, N. Ferro, et al., Seupd@ clef: team iris on temporal evolution of query expansion and rank fusion techniques applied to cross-encoder re-rankers, Faggioli et al.[12] (2024).

[11] S. E. Robertson, U. Zaragoza, The Probabilistic Relevance Framework: BM25 and Beyond, Foundations and Trends in Information Retrieval (FnTIR) 3 (2009) 333–389.

[12] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, et al., The llama 3 herd of models, 2024. URL: https://arxiv.org/abs/2407.21783. arXiv:2407.21783.

[13] Meta, The llama 4 herd: The beginning of a new era of natively multimodal ai innovation, 2025. URL: https://ai.meta.com/blog/llama-4-multimodal-intelligence/, accessed: 2025-05-04.

[14] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2019. URL: http://arxiv.org/abs/1908.10084.

[15] N. Reimers, I. Gurevych, Making monolingual sentence embeddings multilingual using knowledge distillation, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2020. URL: https://arxiv.org/abs/2004.09813.

[16] A. Shakir, D. Koenig, J. Lipp, S. Lee, Boost your search with the crispy mixedbread rerank models, 2024. URL: https://www.mixedbread.ai/blog/mxbai-rerank-v1.

[17] A. E. Elo, The Rating of Chessplayers, Past and Present, Arco Pub., New York, 1978. URL: http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216.

[18] M. Boubdir, E. Kim, B. Ermis, S. Hooker, M. Fadaee, Elo uncovered: Robustness and best practices in language model evaluation, 2023. URL: https://arxiv.org/abs/2311.17295. arXiv:2311.17295.

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

[20] T. Fink, F. Piroi, R. Devaud, P. Galuščáková, G. Gonzalez-Saez, D. Iommi, P. Mulhem, L. Goeuriot, M. Popel, A. El-Ebshihy, Longeval 2025 web retrieval train collection, 2025. doi:10.48436/th5h0-g5f51.

[21] R. Alkhalifa, H. Borkakoty, R. Deveaud, A. El-Ebshihy, L. Espinosa-Anke, T. Fink, P. Galuščáková, G. Gonzalez-Saez, L. Goeuriot, D. Iommi, M. Liakata, H. T. Madabushi, P. Medina-Alias, P. Mulhem, F. Piroi, M. Popel, A. Zubiaga, Overview of the CLEF 2024 LongEval Lab on Longitudinal Evaluation of Model Performance, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction 15th International Conference of the CLEF Association, CLEF 2024, volume 14959 of *Lecture Notes in Computer Science*, Philippe Mulhem and Lorraine Goeuriot and Georges Quénot and Didier Schwab, Springer Nature Switzerland, Grenoble, France, 2024, pp. 208–230. URL: https://hal.science/hal-04737927. doi:10.1007/978-3-031-71908-0_10.

[22] M. Cancellieri, A. El-Ebshihy, T. Fink, P. Galuščáková, G. Gonzalez-Saez, L. Goeuriot, D. Iommi, J. Keller, P. Knoth, P. Mulhem, F. Piroi, D. Pride, P. Schaer, Overview of the CLEF 2025 LongEval Lab on Longitudinal Evaluation of Model Performance, in: J. Carrillo-de Albornoz, J. Gonzalo, L. Plaza, A. García Seco de Herrera, J. Mothe, F. Piroi, P. Rosso, D. Spina, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Sixteenth International Conference of the CLEF Association (CLEF 2025), 2025.