SEUPD Cobalt at CheckThat! 2026: Exploring Sparse and Neural Retrieval from Social Media Claims to Scientific Sources

Notebook for the CheckThat! Lab at CLEF 2026

Filippo Battisti[1], Igor Da Silva[1], Matteo Trevisan[1], Milos Trifunovic[1], Tommaso Zogno[1] and Nicola Ferro[1]

[1]*University of Padua, Italy*

Abstract

This paper describes the system developed by Team Cobalt for the CheckThat! Lab 2026 Task 1: *Source Retrieval for Scientific Web Claims*. The task requires retrieving original scientific publications from a pool of 10,000 candidates based on implicit, multilingual social-media references in English, French and German. Motivated by the working notes of previous editions, where dense neural systems consistently outperformed sparse-only baselines, we pursued two retrieval tracks in parallel. The first is a sparse pipeline built on Apache Lucene and a Python micro-service, designed to be reproducible on commodity hardware: it integrates targeted query preprocessing, synonym compression, MarianMT machine translation, multi-field BM25 scoring (with field weights and pseudo-relevance feedback parameters tuned via Bayesian hyperparameter optimization with Optuna), and neural re-ranking using a TinyBERT cross-encoder; pseudo-relevance feedback was ultimately dropped from the best configuration as it introduced query drift. On the test split this configuration reaches a Mean Reciprocal Rank at 5 (MRR@5) of 0.525 on English, 0.414 on German and 0.510 on French. The second is a neural pipeline executed on the departmental GPU cluster, combining a multilingual bi-encoder with contrastive domain fine-tuning and doc2query document expansion (an LLM-based listwise re-ranking stage was designed but could not be evaluated within the available cluster time). The doc2query plus fine-tuned bi-encoder configuration was the strongest variant we were able to evaluate end-to-end and is the run we submitted to the official task, reaching an average MRR@5 of 0.575 on the test split.

Keywords

CLEF, CheckThat!, 2026, Cobalt

1. Introduction

The rapid growth of scientific literature and its increasing coverage in online discourse has created a new challenge: identifying the source paper behind a scientific claim shared on social media. Users frequently reference research findings in tweets and posts without providing explicit citations, making it difficult to verify the original source.

This work presents the system developed by Team Cobalt at the University of Padua for Task 1 of the CLEF 2026 CheckThat! Lab [1], *Source Retrieval for Scientific Web Claims* [2]. Given a social media post containing an implicit reference to a scientific paper, the task is to retrieve the cited paper from a pool of 10,000 candidate publications. The challenge is compounded by the multilingual nature of the queries, written in English, German and French, and by the informal, noisy language typical of social media.

Previous editions of the task made it apparent that competitive systems consistently relied on dense neural retrieval and learned re-ranking, with sparse-only baselines lagging behind by a substantial margin [2]. Reaching a comparable level of effectiveness with purely sparse techniques was therefore

CLEF 2026: Conference and Labs of the Evaluation Forum, September 21–24, 2026, Jena, Germany

filippo.battisti@studenti.unipd.it (F. Battisti); igorjavier.dasilvalopez@studenti.unipd.it (I. Da Silva); matteo.trevisan.11@studenti.unipd.it (M. Trevisan); milos.trifunovic@studenti.unipd.it (M. Trifunovic); tommaso.zogno@studenti.unipd.it (T. Zogno); nicola.ferro@unipd.it (N. Ferro)

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

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

unlikely. We chose to take this observation seriously from the very beginning of the project, and to develop *two retrieval tracks in parallel*:

- a **sparse track**, built on Apache Lucene and BM25 [3, 4], whose role was to provide a strong, reproducible and lightweight baseline that could run end-to-end on the consumer hardware available to the team;
- a **neural track**, exploring dense bi-encoders, contrastive fine-tuning, document expansion via doc2query, and LLM-based re-ranking [5, 6], executed on the departmental GPU cluster.

The sparse track combines multi-field weighted search, query pre-processing tailored to social-media language, multilingual translation, synonym compression, and neural re-ranking with a TinyBERT cross-encoder (pseudo-relevance feedback was explored but ultimately dropped from the best configuration due to query drift), and achieves an MRR@5 score of 0.525 for English, 0.414 for German and 0.510 for French on the test split. It served both as a strong, reproducible baseline and as a probe to validate design decisions (most notably the inclusion of a cross-encoder re-ranking stage) that were then revisited on the cluster.

The neural track was the system actually selected for the official submission. Among the configurations we were able to evaluate end-to-end, the strongest one combines a multilingual bi-encoder fine-tuned on the task with doc2query document expansion of the abstracts, reaching an average MRR@5 of 0.575 on the test split and clearly improving over the sparse pipeline. A more elaborate variant adding LLM-based listwise re-ranking was designed but, due to limited cluster wall-time at submission deadline, could not be executed in time; it remains the natural starting point for future work, where the GPU cluster will be the standard execution environment.

The paper is organized as follows: Section 2 reviews related work; Section 3 describes our approach, covering both the sparse pipeline and the successive variants of the parallel neural track; Section 4 explains our experimental setup; Section 5 discusses our main findings; Section 6 provides a statistical analysis; finally, Section 7 draws some conclusions and outlooks for future work.

2. Related Work

The task of retrieving scientific literature to verify claims made on social media intersects several active areas of research in Information Retrieval (IR) and Natural Language Processing (NLP), primarily focusing on fact-checking, cross-lingual retrieval, and neural re-ranking architectures.

Fact-Checking and Claim Verification. The rapid proliferation of misinformation on social platforms has driven significant research into automated fact-checking and claim verification [7]. Evaluation campaigns, such as the CLEF CheckThat! Lab [1, 8], have established standardized benchmarks for detecting check-worthy claims and retrieving relevant evidence from large corpora; the specific setting we address, retrieving the scientific source behind a web claim, is the object of Task 1 of the 2026 edition [2]. Unlike general web search, retrieving evidence for scientific claims requires bridging a massive stylistic gap between the informal, often noisy language of social media and the highly technical, formal vocabulary of academic literature.

Vocabulary Mismatch and Sparse Retrieval. Traditional sparse retrieval models, such as BM25 [4], rely heavily on exact lexical matching and consequently struggle with vocabulary mismatch. To mitigate this, query expansion techniques, particularly Pseudo-Relevance Feedback (PRF), have been extensively studied to enrich initial queries with contextual terms extracted from top-ranked documents [9, 10]. Furthermore, text normalization approaches, including stemming [11] and synonym compression, are standard practices to align variant surface forms. However, aggressive normalization can sometimes degrade precision in specialized domains by truncating critical scientific terminology.

Cross-Lingual Information Retrieval (CLIR). When queries and target documents are authored in different languages, the vocabulary mismatch becomes absolute. CLIR approaches typically rely on either translating the query into the document's language, translating the documents, or mapping both into a shared semantic space [12]. Machine translation-based pipelines remain highly effective for cross-lingual retrieval, acting as a robust normalization step, provided the underlying translation models can adequately handle domain-specific entities.

Neural Information Retrieval. The advent of pre-trained language models has revolutionized IR by enabling deep semantic matching that surpasses exact lexical overlap [5]. Modern architectures frequently employ a two-stage retrieve-and-rerank pipeline: a fast first-stage retriever (either a sparse BM25 index or a dense bi-encoder) identifies a candidate pool, which is subsequently re-scored by a computationally expensive, yet highly accurate, cross-encoder [5]. Additionally, document expansion paradigms attempt to predict potential queries a document might answer, effectively translating formal academic prose into closer approximations of user queries prior to indexing, thereby bridging the stylistic gap at the source [13].

3. Methodology

This section details the architecture and implementation of our two retrieval systems. Although they differ fundamentally in their retrieval mechanism and scoring function, both tracks share the same data parsing, query pre-processing and evaluation components. We first describe these shared components together with the sparse pipeline (Sections 3.2–3.8), and then the parallel neural track (Section 3.9), presented as a progressive sequence of variants **N1**–**N6**.

3.1. System Overview

The sparse system is built on top of Apache Lucene [3], an open-source Java library for full-text indexing and retrieval. It is complemented by a Python Flask micro-service (`affinity_service.py`) that handles neural re-ranking [5] and machine translation [12]. The two components communicate via HTTP. The overall pipeline is divided into an offline phase (parsing and indexing) and an online phase (query processing and search).

3.2. Parsing

The collection is stored in a JSON file (`collection_data.json`) where each entry represents a scientific publication with five fields: `pubkey`, `title`, `abstract`, `venue`, and `authors`.

The `CollectionParser` class reads this file incrementally, mapping each entry to a `Document` object via the Jackson library.

The query sets are similarly stored in JSON files (e.g. `en_train.json`), where each entry contains a numeric index, the raw post text, and the ground-truth `pubkey`. These are parsed by the `QuerySolutionParser` class into `QuerySolution` objects.

3.3. Analysis

The `Analyzer` class extends Lucene's standard analyzer with three configurable components: a tokenizer, an optional stemmer, and an optional stop-word and synonym filter.

The tokenizer splits text into tokens, which are then lowercased and filtered against a custom stopword list (`stopwords.txt`). No stemmer is applied in the final system configuration: preliminary experiments with Porter stemming consistently degraded MRR@5 because aggressive truncation collapsed scientific terminology, acronyms and specific medical entities onto unrelated stems, harming precision more than it helped recall [11]. A more detailed analysis is reported in Section 5.

The synonym filter is built from an *explicit-mapping Solr* file (`synonyms.txt`) using Lucene's `SolrSynonymParser` with `expand=false` [14]. Rather than relying on an external thesaurus, we constructed the mapping by prompting a Large Language Model to extract the most frequently-used informal words and slang in the English training set that referenced specific scientific terminology. This collapses variant surface forms to a single canonical English token at index time, reducing vocabulary mismatch between informal post language and formal academic terminology. For example, the mappings collapse `covid-19`, `sarscov2` and `wuflu` to `covid`, and `jab` or `vax` to `vaccination`.

3.4. Indexing

The `Indexer` class writes a Lucene index using BM25 as the similarity function [4]. Each publication is indexed in four separate fields with stored values: `title`, `abstract`, `venue`, and `authors`. Separating the fields allows the searcher to assign different relevance weights to each one.

Before indexing, the `title` and `abstract` of each document are translated to English by the `EnglishTranslator` class. This normalization step is strictly necessary for the sparse pipeline: because BM25 relies on exact lexical matching, leaving queries and documents in their original languages would result in almost zero vocabulary overlap, and thus an extremely low retrieval score, whenever a cross-lingual match is required [12]. To prevent this severe drop in recall, the system projects all texts into a shared English representation.

The translator detects the source language using the Lingua library and forwards non-English text to a local server running Helsinki-NLP MarianMT models for the three secondary languages observed in the collection: French, German, and Spanish. While strictly necessary for cross-lingual recall, a notable limitation of this approach is the substantial increase in offline indexing time, introduced by the neural machine translation bottleneck.

3.5. Query Processing

Before being issued against the index, each query goes through two steps.

First, the `EnglishTranslator` detects the language of the post and translates it to English if necessary, ensuring that both the query and the index share a common language representation.

Second, the `QueryRewriter` applies a sequence of cleaning rules designed to handle the informal nature of social media posts:

- domain-relevant emoji are replaced with their textual equivalents (e.g. the syringe emoji → `vaccine`, the microbe emoji → `virus`);
- HTTP(S) links and anonymous `@user` mentions are removed;
- PascalCase tokens are split at case boundaries (e.g. `SARSCoV2` → `SARS CoV 2`);
- hashtags, underscores, and residual Unicode symbols are removed.

3.6. Searching

The `Searcher` class issues a BM25 query [4] against the Lucene index using a `SimpleQueryParser` configured with per-field boost weights:

- `title` $= 2.0$.
- `abstract` $= 1.8$.
- `authors` $= 1.5$.
- `venue` $= 1.0$.

These weights were chosen heuristically and later validated by hyperparameter optimization (Section 3.8). The ordering reflects the intuition that titles are highly discriminative for this task, abstracts carry most of the topical signal, author names occasionally appear verbatim in the queries, and venue tokens are largely uninformative across a single-domain collection. As a result, a match in the title contributes more to a document's score than an equal-quality match in the abstract.

3.6.1. Pseudo-Relevance Feedback

After the initial BM25 retrieval, the system optionally applies pseudo-relevance feedback (PRF) [9, 10]. The top-k retrieved documents are treated as pseudo-relevant, and the m most frequent terms from their abstracts, excluding stopwords and tokens shorter than three characters, are appended to the query with a reduced boost of 0.5; the expanded query is then re-issued against the index. We start from a deliberately conservative expansion, $k = 5$ pseudo-relevant documents and $m = 10$ expansion terms. Because each query in this task has a single relevant document, a large feedback set is very likely to be dominated by non-relevant abstracts, so both parameters are kept small to limit topic drift, following common practice for short, high-precision queries [9]. These values, together with the multi-field weights, were subsequently refined by the Bayesian optimization of Section 3.8; as reported in Section 5, however, PRF degraded effectiveness for every setting we tried and was therefore excluded from the final configuration.

3.6.2. Neural Re-ranking

The documents retrieved by BM25 are re-ranked by a cross-encoder model (`cross-encoder/ms-marco-TinyBERT-L-2-v2`) served by the Python Flask micro-service [5]. For each candidate, the concatenation of its translated `title` and `abstract` is paired with the cleaned, translated query and scored by the cross-encoder.

Because BM25 scores are theoretically unbounded and highly query-dependent, while the cross-encoder outputs logits on a distinctly different scale, directly combining raw scores can lead to scaling imbalances that disproportionately favor one model. To ensure a stable and proportionate interpolation, we apply per-query Min-Max normalization. For a given query, the raw score s for a candidate document is normalized against the minimum and maximum scores within that specific query's retrieved candidate set S:

$$s' = \frac{s - \min(S)}{\max(S) - \min(S)}$$

Both the BM25 scores and the TinyBERT logits are independently scaled to a strictly $[0, 1]$ range using this method. The final ranking score for each document is then computed as a convex combination of these normalized values:

$$s_{\text{final}} = \alpha \cdot s'_{\text{BM25}} + (1 - \alpha) \cdot s'_{\text{TinyBERT}}$$

The documents are then re-sorted by s_{final}.

The interpolation weight $\alpha = 0.4$ was set heuristically: Bayesian optimization over this parameter was originally planned (Section 3.8), but it proved computationally infeasible on the CPU-only environment used for the sparse pipeline, as discussed in Section 5. Retaining a non-zero weight on both signals keeps the exact-match lexical evidence of BM25 in the final ranking rather than deferring entirely to the neural scorer.

The choice of TinyBERT, rather than a larger and arguably more accurate cross-encoder, was driven by the requirement that the sparse pipeline remain executable end-to-end on a single CPU laptop within a reasonable time budget.

3.7. Evaluation

The `MrrEvaluator` class computes MRR@5 [15] by iterating over all queries, issuing a search against the index, and accumulating the reciprocal rank of the ground-truth publication. The `TrecRunGenerator` class produces a run file in standard TREC format [16] and a corresponding qrels file, which can be used with external evaluation tools.

3.8. Hyperparameter tuning

To maximize the retrieval effectiveness of the proposed pipeline, we integrated a Bayesian hyperparameter optimization module in Python using the Optuna framework. After the manual heuristics, this

approach allowed us to systematically explore the parameter space for the multi-field BM25 weights and the PRF expansion variables. Optimization of the interpolation factor α was also attempted but proved computationally infeasible on the CPU-only environment, as discussed in Section 5. The process was guided by MRR@5 as the target objective function to maximize. For each distinct component of the retrieval pipeline, Optuna executed a sequential series of trials, evaluating the model's performance on the query set to converge toward the optimal configuration that maximizes the chosen metric.

3.9. Parallel Neural Retrieval Track

As anticipated at the beginning of this section, alongside the sparse pipeline we developed an entirely neural track, executed on the departmental GPU cluster. The track was structured as a sequence of progressively richer variants, each motivated by the diagnostic findings of the previous one. Across all variants we reused the parsing, query cleaning and evaluation components of the sparse pipeline, so that the comparison between configurations isolates the effect of the retrieval and re-ranking models. Throughout this section, the configurations are labelled with descriptive names (**N1** through **N6**), which are also the labels used in the corresponding result table in Section 5.

3.9.1. N1: off-the-shelf multilingual bi-encoder

The first variant replaced sparse BM25 with `BAAI/bge-m3`, a multilingual sentence-transformer bi-encoder [5], used in a zero-shot regime. Each publication was encoded as the concatenation of its title and abstract, each query as the cleaned post. The full collection of 10,000 publications was encoded once offline, and exact nearest-neighbour search on the resulting vectors was performed in memory using FAISS, with cosine similarity as the ranking score.

Two properties of this design motivated the rest of the track. On the one hand, a dense encoder is robust to the vocabulary mismatch that is the most pervasive failure mode of BM25 on this collection, because it maps semantically related expressions to nearby points in the embedding space, and it is natively multilingual, needing no explicit translation step. On the other hand, a general-purpose semantic-similarity encoder is not specialized for the very particular notion of relevance this task imposes, namely *"this informal post implicitly cites this specific scientific paper"*. The first property argued for building on a dense retriever, the second for adapting it to the task, which is the object of **N2**.

3.9.2. N2: bi-encoder with domain fine-tuning

The natural next step was to specialize the bi-encoder on our task. Starting from the same multilingual encoder (`BAAI/bge-m3`, maximum sequence length 512), we fine-tuned it with a contrastive Multiple Negatives Ranking Loss (MNRL) on the (post, gold paper) pairs from the training splits of all three languages, pooled and shuffled together ($\approx 19{,}000$ pairs). As in **N1**, each document is represented by the concatenation of its title and abstract.

Fine-tuning proceeds in two stages, so that hard negatives are mined by the model itself once it has already acquired a first notion of task relevance:

1. *Warm-up* (1 epoch). We optimize MNRL with *in-batch* negatives only: within each batch of 256 (post, gold paper) pairs, the 255 gold papers of the other queries act as easy negatives. A cached MNRL implementation splits the logical batch into physical mini-batches of 64 to fit GPU memory, with 200 linear warm-up steps.
2. *Hard-negative fine-tuning* (2 epochs). Using the warm-up encoder, for every query we retrieve its 30 nearest documents by cosine similarity and keep the 5 highest-ranked ones that are *not* the gold paper as explicit hard negatives. Each training example is therefore a tuple of one query, one positive and five mined hard negatives (seven texts), while MNRL additionally reuses the in-batch positives of the other queries as further negatives. This stage uses a logical batch of 96 (physical mini-batch 16) and 100 warm-up steps.

Mining the hard negatives from the warm-up encoder itself, rather than from BM25, keeps the distractors on the same semantic manifold on which the model is being optimized and yields negatives that are genuinely difficult for a dense retriever. All runs use a fixed random seed (42), and model selection uses MRR@5 on the development split; the complete hyperparameter configuration is released with the code.

3.9.3. N3: N2 + generic cross-encoder re-ranker

Following the standard two-stage retrieve-then-rerank paradigm, this variant adds a cross-encoder on top of the fine-tuned bi-encoder: the top-100 candidates returned by dense retrieval are re-scored by a strong off-the-shelf multilingual cross-encoder (`BAAI/bge-reranker-v2-m3`, input length 512), and the final ranking is obtained by sorting on the cross-encoder score. This is the textbook way to sharpen the ordering of a first-stage retriever, and our working hypothesis was that it would refine the top of the **N2** ranking; whether it actually does is examined in Section 5.

3.9.4. N4: N2 + cross-encoder fine-tuned on the task

To test whether the regression of **N3** was merely due to the re-ranker's lack of domain adaptation, we fine-tuned the same cross-encoder (`BAAI/bge-reranker-v2-m3`) on the (post, gold paper) pairs, using hard negatives re-mined with the fine-tuned bi-encoder of **N2** (2 epochs, batch size 32, learning rate 2×10^{-5}), and used it as the second-stage scorer on top of **N2**. If the re-ranker's out-of-domain training signal were the only problem, this domain-adapted variant should recover the lost effectiveness.

3.9.5. N5: doc2query expansion + N2 (submitted run)

We then turned to the document side. Each abstract was expanded, offline, with three synthetic queries generated by a multilingual docTTTTTquery model (`doc2query/msmarco-14langs-mt5-base-v1`) [13] using nucleus sampling (top-$k = 50$, top-$p = 0.95$). The intuition is that an abstract describes a paper in the language of academic prose, whereas a query is phrased as an informal social-media claim; doc2query produces, for every document, plausible queries it is expected to answer, and appending them to the indexed text narrows this stylistic gap at the source. The expanded corpus is then encoded by the fine-tuned bi-encoder of **N2**. This is the run we submitted to the official task; its effectiveness is reported in Section 5.

3.9.6. N6: doc2query + N2 + fine-tuned cross-encoder + RRF with sparse

For completeness, we also assembled a full hybrid. Starting from **N5**, we re-introduced the fine-tuned cross-encoder of **N4** as a second stage and fused the dense ranking with the sparse BM25 ranking through Reciprocal Rank Fusion (RRF, $k = 60$) [17] before re-ranking the top-100 fused candidates. The rationale is that dense and sparse retrievers should be complementary, so combining them and then re-ranking might yield the best of both worlds; whether the added components help is discussed in Section 5.

3.9.7. Designed but not evaluated: LLM-based listwise re-ranking

A further variant was designed and partially implemented, in which the cross-encoder re-ranking stage of **N6** would have been replaced by an LLM-based listwise re-ranker [6]. The top candidates returned by **N5** would have been presented to an instruction-tuned large language model with a listwise prompt that asked the model to identify, given the post, which of the candidate papers was most likely the source. The LLM would have produced a complete ordering of the candidates, used directly as the final ranking, with the bi-encoder score as a tie-breaker. The expected advantage over a similarity-based reranker is the LLM's ability to perform an implicit reasoning step ("the post claims that vaccine X

causes side effect Y, does this paper actually report on X and Y?") that no similarity scorer can carry out reliably.

Owing to limited cluster wall-time available before the submission deadline, we were unable to run this configuration end-to-end on the development or test splits, and we therefore do not report numbers for it. We mention it here because it is the natural starting point for future work: a working LLM listwise re-ranking stage on top of the doc2query plus fine-tuned bi-encoder retriever is the direct continuation of the neural track described above.

4. Experimental Setup

This section describes the experimental framework: the dataset, the evaluation metric, and the two hardware environments (commodity laptops for the sparse track and a GPU cluster for the neural track) used to obtain the results reported in Section 5.

4.1. Collection

The document collection used for this task is the CheckThat! Lab 2026 Task 1 dataset, curated for the challenge *Source Retrieval for Scientific Web Claims*. It consists of a pool of 10,000 scientific publications, written in four different languages: English, French, German and Spanish. Each document is described by five structured fields: a unique identifier (`pubkey`), `title`, `abstract`, `venue`, and `authors`.

The query sets are *social media posts* written in English, German, and French, each implicitly referencing one of the publications in the pool. Importantly, the query sets contain cross-lingual references (e.g. English queries referencing German documents, and vice versa). Furthermore, the English query set also contains Spanish queries, thus introducing noise during retrieval tasks. For each language, separate training and development splits are provided (e.g. `en_train.json`, `en_dev.json`), as well as a held-out test set (`final_en_test.json`). Each entry contains a numeric index, the raw post text, and (in the training and development sets) the ground-truth `pubkey` of the cited paper.

4.2. Evaluation Measures

The primary evaluation metric adopted for this task is **Mean Reciprocal Rank at cut-off 5 (MRR@5)**. Given a set of queries Q, it is defined as [15]:

$$\mathrm{MRR@5} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\mathrm{rank}_q}$$

where $\mathrm{rank}_q \in \{1, 2, 3, 4, 5\}$ is the position of the first relevant document (the ground-truth `pubkey`) in the top-5 results for query q; the term is 0 if no relevant document appears within the first five ranks [15]. MRR@5 rewards systems that place the correct paper as high as possible in the ranked list, reflecting the typical user behaviour of inspecting only the first few results.

4.3. Hardware

Two distinct execution environments were used in this work, reflecting the dual-track design described in Section 3.

The **sparse track** was developed and evaluated on the personal laptops of the group members, which represent a realistic "commodity hardware" deployment scenario. The most constrained configuration in the team is an ASUS notebook with an Intel Core i5-1035G1 CPU (1.00 GHz base, up to 3.60 GHz turbo) and 4 GB of RAM; the remaining team machines are comparable consumer laptops without dedicated GPUs. All numbers reported for the sparse pipeline, including the TinyBERT re-ranking stage, were obtained on this kind of hardware.

The **neural track** was instead executed on the departmental GPU cluster, which provides access to NVIDIA GPUs with sufficient memory to host multilingual sentence-transformer encoders, doc2query

generation models, and instruction-tuned LLMs used for listwise re-ranking. Bulk embedding of the 10,000-publication collection, contrastive fine-tuning of the bi-encoder, and LLM-based re-ranking would have been impractical on the consumer hardware available to the team, and were therefore restricted to the cluster.

5. Results and Discussion

This section evaluates the two retrieval tracks, reporting all numbers as Mean Reciprocal Rank at cut-off 5 (MRR@5) on the test split. We present the sparse and neural tracks in turn (Section 5), compare their computational cost (Section 5.2), and then discuss the phenomena behind the numbers: synonym compression, cross-lingual drift, the harmful effect of PRF, and the surprisingly negative effect of cross-encoder re-ranking on top of an already specialized bi-encoder.

5.1. Experimental Results

5.1.1. Sparse Pipeline

We initially established a baseline using BM25 retrieval [4] over a single-field index (concatenating the title, authors, and abstract). With standard tokenization, stopword removal, and no stemming, this baseline achieved an MRR@5 of 0.437 for English, while the cross-lingual performance for French and German lagged significantly at around 0.083 and 0.071. This low score was primarily due to the severe vocabulary mismatch between the noisy, multilingual social media queries and the formal, academic index.

To address these shortcomings, we incrementally augmented the system through a heuristic pipeline. We introduced multi-field weighted search to prioritize title matches, followed by query pre-processing and synonym compression to normalize the social media language. Multilingual translation was then applied, yielding a significant performance jump to an MRR@5 of 0.489 for English, 0.358 for German and 0.471 for French. Finally, with the addition of Pseudo-Relevance Feedback (PRF) [9, 10], the performance of the system dropped significantly due to query drift.

Adding TinyBERT neural re-ranking [5] on top of the PRF stage improved the scores, but they remained below the pre-PRF configurations. A post-hoc evaluation on the test split (**MQSTR**, reported in Section 6) shows that applying the same re-ranker directly on the clean post-translation candidate set, bypassing PRF, recovers the lost performance and yields the strongest sparse configuration on every language, a result consistent with the "PRF paradox" diagnosis: PRF corrupts the candidate set so severely that no amount of downstream re-ranking fully compensates.

Consequently, by discarding the PRF step, the system achieved a peak performance of 0.525 for English, 0.414 for German, and 0.510 for French.

In parallel, we subjected the architecture to Bayesian hyperparameter optimization via the Optuna framework [18], which identified improved parameters for the multi-field BM25 weights and PRF variables; optimization of the re-ranking interpolation weight α proved computationally infeasible on the CPU-only environment due to critical memory swapping bottlenecks, as discussed below.

5.1.2. Parallel Neural Track

On the cluster, the six fully evaluated variants of the neural track produced a coherent picture, summarized in Table 2 and discussed below. The off-the-shelf multilingual bi-encoder (**N1**) was already a clear improvement over the un-engineered single-field BM25 baseline, in particular on French and German, but remained below the heuristic sparse pipeline.

The single largest jump was produced by domain fine-tuning of the bi-encoder (**N2**), which raised the average MRR@5 to 0.572 and overtook the sparse pipeline by a wide margin on every language split. Adding a generic cross-encoder re-ranker on top (**N3**) regressed performance to an average of 0.536, confirming that a re-ranker trained on out-of-domain web passages is not a useful complement

Table 1
MRR@5 on the **test split** per language for the evaluated configurations of the sparse pipeline. The heuristic pipeline shows the incremental impact of each architectural component. Per-query traces, on which Section 6 runs the significance tests, are released under `results/mrr5s/`.

Configuration	MRR@5 en	MRR@5 de	MRR@5 fr
BM25 baseline (single field, no pre-processing)	0.437	0.083	0.071
Heuristic Pipeline (Incremental)			
+ Multi-field weighted search (**M**)	0.450	0.085	0.075
+ Query pre-processing (**MQ**)	0.487	0.091	0.076
+ Synonym compression (**MQS**)	0.489	0.089	0.076
+ Multilingual translation (**MQST**)	0.489	0.358	0.471
+ Pseudo-relevance feedback ($K = 5$, $M = 10$) (**MQSTP**)	0.308	0.203	0.269
+ TinyBERT re-ranking ($\alpha = 0.4$) (**MQSTPR**)	0.468	0.358	0.445
+ TinyBERT re-ranking (w/o PRF) ($\alpha = 0.4$) (**MQSTR**)	**0.525**	**0.414**	**0.510**

to a retriever already specialized on the task. Surprisingly, re-ranking with a cross-encoder fine-tuned on the same (post, gold paper) pairs (**N4**) did not recover this loss but regressed further to an average of 0.515, falling below both **N3** and the bi-encoder retriever on every language split.

The strongest fully evaluated configuration was **N5**, which adds doc2query document expansion on top of the fine-tuned bi-encoder. This is the run we eventually submitted to the official task. Doc2query expansion provided a small additional gain over **N2**, raising the average MRR@5 to $\mathbf{0.575}$, with the largest contribution on French (MRR@5 $= 0.606$). The improvement over **N2** (0.572) is small in aggregate and the two configurations are statistically indistinguishable on the per-query test (Section 6), but **N5** is consistently at or above **N2** on every language. Finally, re-introducing a fine-tuned cross-encoder and a Reciprocal Rank Fusion (RRF) stage with sparse BM25 on top of **N5** (configuration **N6**) once again regressed performance to an average of 0.435, with a particularly dramatic collapse on German (down to 0.319), in line with the consistent observation that on this collection a re-ranker on top of an already strong dense retriever is harmful rather than helpful.

A further variant replacing the cross-encoder re-ranker with an LLM-based listwise re-ranker on top of **N5**, described in Section 3.9.7, was designed but could not be evaluated within the cluster wall-time available before the submission deadline. We do not report numbers for it and we leave its end-to-end evaluation as the immediate starting point for future work.

Table 2
MRR@5 on the **test split** per language for the six fully evaluated configurations of the parallel neural track, and their average. **N5** is the configuration that we submitted to the official task. Per-query MRR@5 traces, used by the significance tests in Section 6, are released under `results/mrr5s/`.

Configuration	en	de	fr	avg
N1: off-the-shelf bi-encoder	0.493	0.406	0.478	0.459
N2: bi-encoder, domain fine-tuned	**0.578**	0.540	0.600	0.572
N3: **N2** + generic cross-encoder re-ranker	0.558	0.478	0.573	0.536
N4: **N2** + cross-encoder fine-tuned on the task	0.538	0.464	0.543	0.515
N5: doc2query + **N2** *(submitted)*	**0.578**	**0.541**	**0.606**	**0.575**
N6: doc2query + **N2** + fine-tuned re-ranker + RRF (sparse)	0.558	0.319	0.428	0.435

5.2. Computational Cost

Effectiveness is only half of the picture: the neural variants differ sharply in their computational footprint, and the cheapest configuration turns out to be the most effective one. Table 3 reports the measured cost of each component, separating the *one-time, offline* cost (paid once, at training or indexing

time) from the *online* cost paid per query at retrieval time. All neural figures were measured on a single NVIDIA A40 (48 GB); the sparse pipeline runs entirely on the commodity CPU laptops of Section 4.

Table 3
Measured computational cost of the retrieval components. One-time costs are paid once (training or offline indexing); online costs are per query at retrieval time. Neural figures are on a single NVIDIA A40 (48 GB) over the 10,000-document collection; the dense retrieval cost is amortized over the batch-encoded corpus.

Component (variant)	Hardware	One-time	Per query
Bi-encoder fine-tuning (**N2**)	A40	$\approx$2.2 h	–
doc2query corpus expansion (**N5**)	A40	$\approx$5 min	–
Corpus encoding (per run)	A40	$\approx$1–3 min	–
Dense retrieval (**N1**/**N2**/**N5**)	A40	–	$<$1 ms
Cross-encoder fine-tuning (**N4**)	A40	$\approx$1.6 h	–
Cross-encoder re-rank, top-100 (**N3**/**N4**/**N6**)	A40	–	$\approx$2.3 s
Sparse pipeline + TinyBERT re-rank	CPU laptop	translation-bound	CPU-only

The practical takeaway reinforces the effectiveness ranking. Once the bi-encoder has been fine-tuned (a one-time cost of about two hours) and the corpus has been encoded, dense retrieval is essentially free at query time (sub-millisecond per query over an in-memory index of 10,000 vectors). doc2query adds only a few minutes of one-time offline expansion and leaves the online cost unchanged. In contrast, every cross-encoder re-ranking stage (**N3**, **N4**, **N6**) adds roughly 2.3 s per query, three orders of magnitude more than the dense retriever it sits on top of, and **N4** further requires about 1.6 h of re-ranker fine-tuning, yet Table 2 shows that these stages *lower* effectiveness. The submitted **N5** is therefore not only the most accurate variant we evaluated but also, at query time, among the cheapest.

5.3. Discussion

Query Pre-processing and Synonym Compression. Targeted cleaning rules, such as splitting PascalCase hashtags and mapping domain-relevant emojis to text (e.g. the syringe emoji to "vaccine"), together with a synonym filter that collapses informal slang ("jab", "wuflu") onto its canonical academic form ("vaccination", "covid"), consistently improved retrieval accuracy on the sparse side by reducing the lexical gap.

Stemming. Porter stemming degraded highly technical queries, truncating scientific terminology, acronyms and medical entities onto unrelated stems and lowering precision [11]; it was therefore disabled in the final configuration.

Translation Quality. MarianMT produces fluent French and German translations, but occasional errors on scientific terms (e.g. drug names or gene symbols) can mislead retrieval [12]; a domain-adapted model or a glossary-constrained decoder could mitigate this in future iterations.

The Alpha Parameter. On the sparse side the interpolation weight was fixed heuristically at $\alpha = 0.4$. Optimizing it via Optuna proved infeasible on the CPU-only environment, where the Bayesian search triggered severe memory swapping, so $\alpha = 0.4$ remains our operating point; automating this search is left to the GPU-based follow-up.

Multilingual Queries Referencing English Titles. A fraction of French and German queries cite English paper titles verbatim, where translation occasionally hurt retrieval because the translated form no longer matched the exact indexed string. A hybrid query that keeps the original text alongside its translation could address this edge case.

Why fine-tuning was decisive in the neural track. The single largest jump in the neural track was not produced by any re-ranking stage but by domain fine-tuning of the bi-encoder (**N1** to **N2**). This is consistent with the observation that the underlying retrieval task, aligning a noisy informal claim with the specific paper that supports it, is sufficiently far from the generic semantic-similarity objective on which sentence-transformers are pre-trained that an off-the-shelf encoder can only partially capture it [5]. The same observation explains why **N3** regressed: a cross-encoder trained on general-domain retrieval data cannot correct a retriever that has already been specialized on the task, because its training signal is, if anything, less aligned with the desired notion of relevance than the fine-tuned bi-encoder's.

Why even a fine-tuned cross-encoder fails to help. A more surprising finding is that fine-tuning the cross-encoder on the same data used for the bi-encoder did not, on its own, restore the lost performance: on the test split **N4** did not even match the generic re-ranker **N3**, regressing further below the bi-encoder retriever on every language. We attribute this to two structural factors. First, once the first-stage retriever is sufficiently strong, the marginal information that a re-ranker can add is small, and is partly cancelled out by the truncation of long abstracts at the cross-encoder input length. Second, the cross-encoder's hard input-length bound forces it to focus on a narrow slice of each abstract, while the dense bi-encoder pools information from the entire document, which is the type of signal that this collection rewards. Domain fine-tuning of the cross-encoder also amplifies its tendency to commit to a small number of high-scoring candidates, which on this test split turns out to be harmful rather than helpful.

Why doc2query is the right kind of expansion here. The contribution of doc2query (**N5**) is qualitatively different. Rather than scoring an existing candidate set, doc2query *rewrites* each document so that the language used to describe it is closer to the language used in the queries. The synthetic queries appended to each abstract carry exactly the kind of informal, claim-like phrasing that social-media posts use, and this bridges the stylistic gap that no amount of cross-encoder re-ranking seems able to close on its own. The fact that **N5** is also the strongest configuration we submitted is consistent with this diagnosis.

Why stacking complexity hurts. Configuration **N6** adds, on top of **N5**, a fine-tuned cross-encoder re-ranker and an RRF fusion with the sparse BM25 ranking. Despite combining apparently complementary signals, this configuration regresses below **N5** on every language. This mirrors what we already observed in **N3** and **N4**: on this collection, a re-ranker on top of an already strong dense retriever introduces more noise than signal, and RRF with sparse BM25 contaminates a clean dense ranking with the lexical errors that the bi-encoder had already learnt to compensate for [17]. The one re-ranking stage we could not evaluate, an LLM-based listwise re-ranker [6] on top of **N5**, is discussed as future work in Section 7.

6. Statistical Analysis

The aggregate MRR@5 numbers reported in Section 5 suggest a clear ordering of the configurations on each language, but they do not, by themselves, tell us whether the observed differences are systematic or could be explained by query-level variability. We complement Tables 1 and 2 with inferential tests that operate on the per-query MRR@5 traces produced by every system.

The analysis is organized in three families, with the family-wise error rate (FWER) controlled within each family separately at $\alpha = 0.05$:

1. A within-track comparison per language.
2. A cross-track headline comparison that ranks all fourteen pipelines on the same query sample.
3. A factorial analysis of the PIPELINE and LANGUAGE factors and their interaction: a two-way ANOVA on the pipeline×language design, a marginal Tukey HSD that ranks the pipelines across

languages, and a topic-aware nested (split-plot) mixed model that we use as a robustness check on the factorial conclusions.

6.1. Methodology

Data. For every system we have a per-query MRR@5 dump $\mathrm{mrr5}(q) \in \{0, 0.2, 0.25, 0.\overline{3}, 0.5, 1.0\}$ over the test queries. The sparse pipeline (**base**, **M**, **MQ**, **MQS**, **MQST**, **MQSTR**, **MQSTP**, **MQSTPR**) and the neural pipeline (**N1**–**N6**) were both run on the same query set per language. **MQSTR** is the variant that applies the TinyBERT re-ranker directly on top of the post-translation candidate set (**MQST**), bypassing the pseudo-relevance feedback stage; it isolates the contribution of the re-ranker from the well-known candidate-set drift introduced by PRF. Every test below is paired by query identifier: we restrict to the intersection of query IDs across the systems being compared before computing any statistic, so the same set of queries contributes to every cell of a given test family.

Figure 1 gives a high-level visual summary of the per-query outcomes that drive the analysis. For each configuration we show how the queries are partitioned among the six possible MRR@5 outcomes (gold document found at rank 1, 2, 3, 4, 5, or not found in the top-5). This breakdown explains why MRR@5 is better characterised by the proportion of rank-1 hits and the proportion of misses than by quartile-based summaries: the per-query distribution is essentially bimodal at 0 and 1, and a classical boxplot collapses to a thin segment with no useful information.

Choice of tests. MRR@5 is bounded in $[0, 1]$ and heavily mass-bimodal. The normality assumption of parametric tests is therefore strongly violated. Our per-system comparison is the non-parametric Friedman test followed by pairwise Wilcoxon signed-rank tests with Holm–Bonferroni correction; this is the standard procedure for IR system comparisons since the work of [19] and controls the FWER exactly over the $\binom{k}{2}$ pairs of a k-system family.

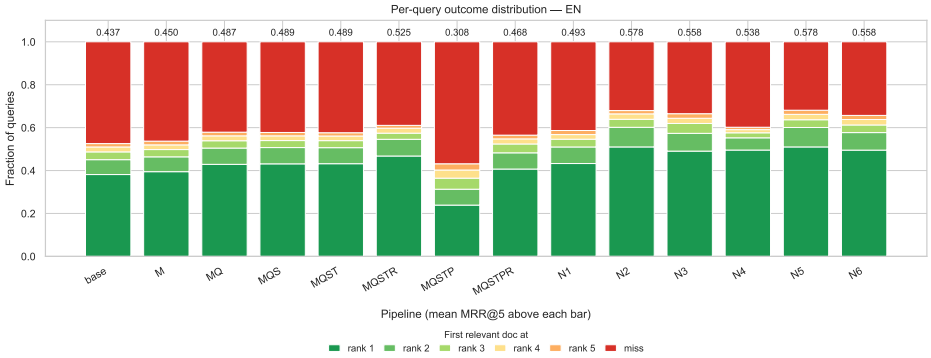
Factorial design and the role of TOPIC. The canonical comparison on a single test collection is a two-way SYSTEM×TOPIC ANOVA with TOPIC as a random blocking factor, since per-topic difficulty dominates the variance and blocking on it is what gives the SYSTEM test its power. A three-way extension is not estimable here: the three languages have *disjoint* query sets, so TOPIC is *nested* within LANGUAGE rather than crossed with it. The natural topic-aware model is therefore the split-plot design $\mathrm{mrr5} \sim \mathrm{SYSTEM} * \mathrm{LANGUAGE} + (1 \mid \mathrm{TOPIC})$, which we fit as a robustness check in Section 6.3.

Our primary per-system inference already blocks on TOPIC: the per-language Friedman test is a randomized-complete-block analysis with the topic as block, adopted because MRR@5 badly violates ANOVA normality. We use the parametric two-way ANOVA only for the pipeline×language *interaction*, which the per-language tests cannot express; it is fitted by ordinary least squares on the long-format data with Type-II sums of squares, appropriate for our strongly unbalanced design (EN 6,076, DE 876, FR 1,220 queries). The PIPELINE marginal effect is summarized by the Tukey HSD of Section 6.3; the LANGUAGE marginal effect is confounded with query-set difficulty and reported with that caveat.

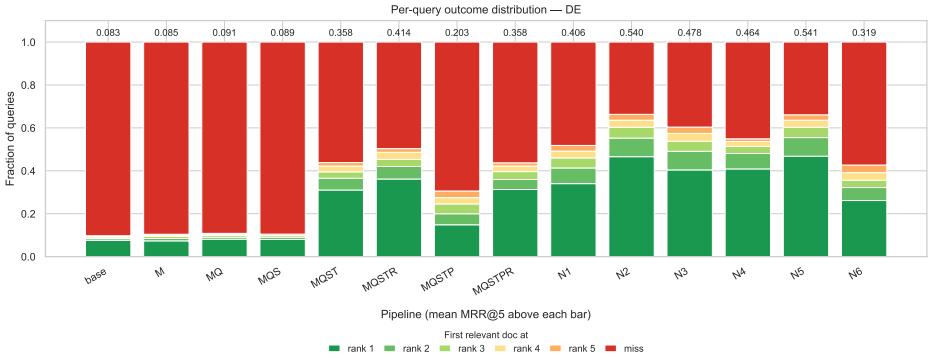
6.2. Per-language results

For each language and each track the Friedman test rejects the null hypothesis that all configurations have the same MRR@5 distribution with overwhelming evidence (smallest $\chi^2 \approx 401$, largest $p \approx 2 \cdot 10^{-84}$). The Holm-corrected pairwise Wilcoxon tests then identify which configurations are pairwise different; Table 4 summarizes the number of significant pairs per family.

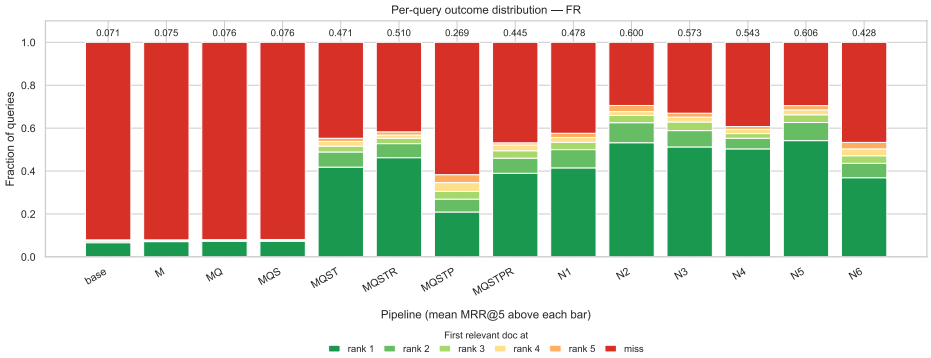
The full pairwise picture is reported in the cross-track Critical Difference diagram of Figure 2 [19], which displays the average rank of each of the fourteen systems across the queries (lower is better) and connects with a horizontal bar those that are *not* significantly different at $\alpha = 0.05$ under the Nemenyi post-hoc. Within the sparse track this places **MQSTR** as the unique EN winner, separated from the **MQ**/**MQS**/**MQST** cluster, and inside the top sparse group on DE/FR; within the neural track **N2** and **N5** are tied at the top, **N3** and **N4** regress below them, and **N6** falls to the bottom on DE and FR.



(a) English



(b) German



(c) French

Figure 1: Per-query outcome distribution. Each bar is partitioned into the fraction of queries that scored MRR$= 1$ (gold at rank 1), MRR$= 0.5$ (rank 2), and so on down to MRR$= 0$ (miss). The mean MRR@5 of the configuration is annotated above each bar. The neural fine-tuned configurations (**N2**–**N5**) systematically dominate the sparse track by converting both "miss" and "mid-rank" outcomes into "rank 1" hits. On DE and FR the gap with the pre-translation sparse configurations (**base**, **M**, **MQ**, **MQS**) is enormous and almost entirely due to the much larger fraction of rank-1 hits. Within the sparse track **MQSTR** (TinyBERT re-rank applied directly on the post-translation candidates, skipping PRF) is the strongest configuration on every language and the only sparse pipeline that converts a non-trivial share of "miss" queries into top-5 hits.

The most important pairwise differences are summarized in Table 5. **N5** is significantly better than **N1** and than every sparse configuration on every language, but its gain over **N2** is *not* significant ($p \geq 0.29$ everywhere), confirming that doc2query is only a marginal addition once the bi-encoder is domain-fine-tuned. The post-**N2** re-ranking strategies **N3**, **N4** and **N6** all regress significantly against **N2**/**N5**, so they hurt the per-query distribution and not merely the aggregate. On the sparse side, adding the re-ranker on top of PRF (**MQSTPR** vs. **MQSTP**) recovers most of the PRF loss but does not catch up with **MQST** on EN, whereas applying the re-ranker directly to the clean post-translation candidate set

Table 4
Number of Holm-significant pairwise differences (Wilcoxon signed-rank, $\alpha = 0.05$) per analysis family. The denominator is $\binom{k}{2}$.

Family	Language	k	Sig. pairs
sparse	EN	8	25/28
sparse	DE	8	21/28
sparse	FR	8	22/28
neural	EN	6	13/15
neural	DE	6	13/15
neural	FR	6	15/15
all 14	EN	14	83/91
all 14	DE	14	79/91
all 14	FR	14	80/91

(**MQSTR**) closes the gap and significantly overtakes both **MQSTPR** and **MQST**. **MQSTR** matches the off-the-shelf bi-encoder **N1** on DE and FR but still trails the domain-fine-tuned **N2**/**N5** by ≈ 0.10–0.13 there.

6.3. Pipeline × language interaction

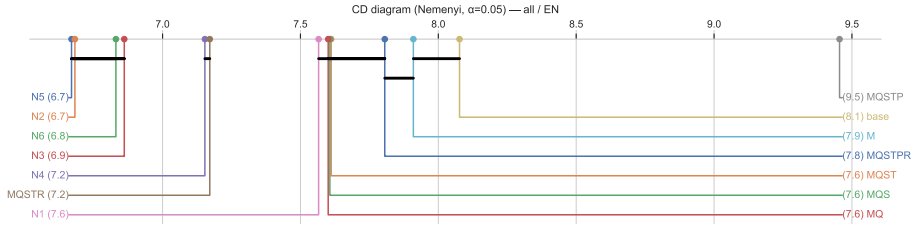
We finally fit a two-way ANOVA on the long-format per-query MRR@5, separately for the sparse and the neural track, with PIPELINE and LANGUAGE as factors and Type-II sum of squares to handle the unbalanced design.

The pipeline×language interaction is large and significant in both tracks, but its origin is qualitatively different. In the sparse track the interaction is driven almost entirely by the multilingual translation stage **T**: switching from **MQS** (no translation) to **MQST** (with translation) raises the mean MRR@5 by $+0.27$ on DE and $+0.39$ on FR but remains unchanged on EN. The downstream re-ranker (**R**: **MQST** → **MQSTR**) adds a smaller, much more uniform gain across the three languages ($+0.036$/$+0.056$/$+0.040$ on EN/DE/FR), confirming that the re-ranker contributes signal of its own once the candidate set is clean. In the neural track the interaction is smaller, and is mostly explained by **N6** (RRF fusion with sparse) hurting DE disproportionately more than EN or FR, while the cross-encoder re-rankers **N3** and **N4** preserve a stable relative ordering across the three languages.

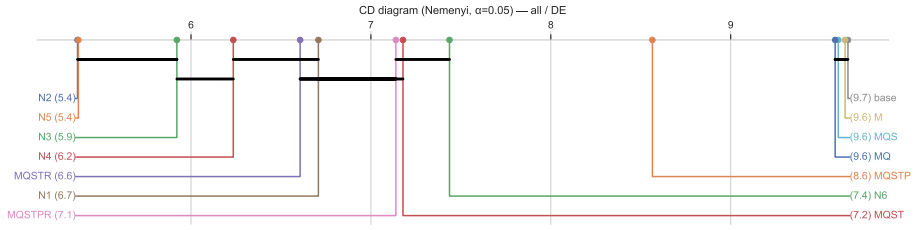
The LANGUAGE marginal effect is highly significant in both tracks, but, as anticipated, it conflates the average difficulty of the three query corpora and should not be read as a statement about single queries being intrinsically easier in one language than another.

Which pipeline is best across languages? Pooling the per-query scores over the three languages and applying a Tukey HSD to the PIPELINE main effect yields a single cross-language ranking with FWER control (Table 7). In the sparse track **MQSTR** is the unique top group, significantly ahead of every other configuration; **MQST** and **MQSTPR** are tied just below it, and **MQSTP** is significantly the worst. In the neural track **N5** and **N2** form the top group and are mutually indistinguishable, **N3** is alone in second, **N4** and **N6** are tied, and **N1** is significantly the worst. This confirms across languages the per-language verdicts of Section 6.2. Two caveats apply: the pooled marginal mean is query-weighted and therefore EN-dominated, so the ordering of the four pre-translation sparse configurations (**base**, **M**, **MQ**, **MQS**) reflects EN, where translation is a no-op; under equal per-language weighting the translation-bearing **MQSTP** overtakes them, a direct manifestation of the large pipeline×language interaction. The top of each ranking (**MQSTR**; **N5**/**N2**) is stable under both weightings.

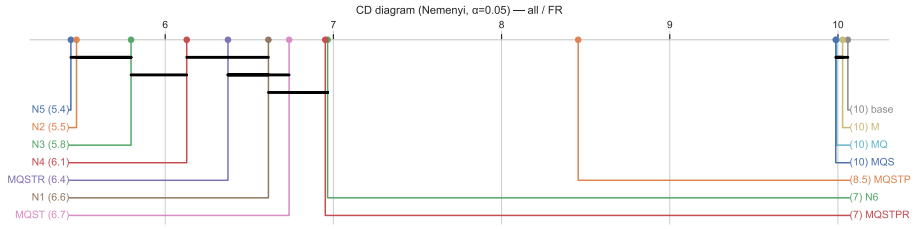
Topic-aware nested model. Because the two-way ANOVA above treats the per-query scores as independent replicates, it leaves the (large) per-topic variability in the residual. As a robustness check



(a) English

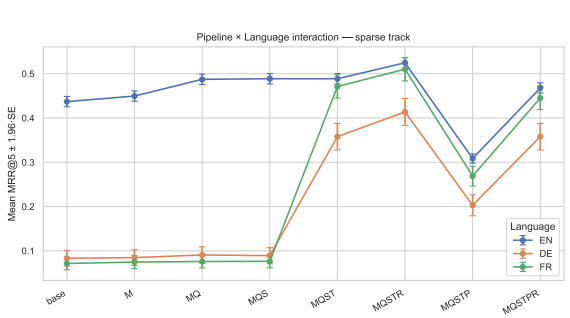


(b) German

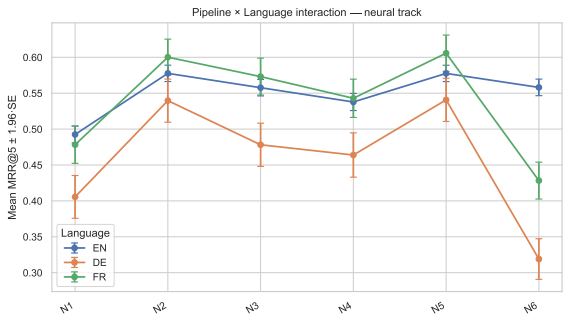


(c) French

Figure 2: Cross-track Critical Difference (Nemenyi, $\alpha = 0.05$) diagram across all fourteen configurations, per language. Horizontal bars connect groups of systems with no significant difference. The top of the ranking on every language is occupied by the fine-tuned neural configurations **N2** and **N5**, which are statistically indistinguishable; the cross-encoder re-ranking variants **N3** and **N4** sit below them. The strongest sparse pipeline is **MQSTR**, which on EN sits in the same Nemenyi clique as **N4** (despite a small, Wilcoxon-significant gap of ≈ 0.015) and overtakes every other sparse configuration on every language. The bottom of the ranking is always **MQSTP** (PRF without re-ranker), which is worse than the BM25 baseline.



(a) Sparse track



(b) Neural track

Figure 3: Pipeline $\times$ language interaction plots. Error bars are $1.96 \cdot$ SE around the per-cell mean MRR@5. The sparse panel shows the strong, non-additive contribution of the multilingual translation (**T**: **MQS** $\rightarrow$ **MQST**): DE and FR jump up by roughly 0.27 and 0.39, while EN is essentially flat. The peak on every language is **MQSTR** (re-rank on the clean post-translation candidates), with a further $+0.04$ to $+0.06$ over **MQST**, re-ranking helps when fed a clean candidate set, but the same **R** step on top of PRF (**MQSTP** $\rightarrow$ **MQSTPR**) only partially recovers from the drift PRF introduces. On the neural side the curves cross more sharply: **N6** hurts DE drastically (down to 0.319, against 0.428 on FR and 0.558 on EN), while among the cross-encoder re-rankers **N3** sits consistently above **N4** on every language without ever closing the gap with **N2**/**N5**.

Table 5
Selected pairwise differences from the cross-track family. p-values are Holm-corrected over the 91 pairs of the family. "mean diff" is signed as $\mu(A) - \mu(B)$; ✓ marks a Holm-significant difference at $\alpha = 0.05$.

Lang	Pair A vs. B	p_{Holm}	mean diff	sig.
EN	N5 vs. N2	1.00	-0.000	–
EN	N5 vs. N1	$2.2 \cdot 10^{-79}$	$+0.085$	✓
EN	N5 vs. MQSTR	$1.2 \cdot 10^{-25}$	$+0.054$	✓
EN	N5 vs. MQSTPR	$1.7 \cdot 10^{-97}$	$+0.111$	✓
EN	N5 vs. N6	$3.0 \cdot 10^{-5}$	$+0.020$	✓
EN	N2 vs. N3	$7.0 \cdot 10^{-5}$	$+0.021$	✓
EN	N2 vs. N4	$6.4 \cdot 10^{-14}$	$+0.040$	✓
EN	MQSTR vs. MQST	$4.1 \cdot 10^{-37}$	$+0.036$	✓
EN	MQSTR vs. MQSTPR	$3.2 \cdot 10^{-84}$	$+0.057$	✓
EN	MQSTPR vs. MQSTP	$2.9 \cdot 10^{-260}$	$+0.159$	✓
DE	N5 vs. N2	1.00	$+0.001$	–
DE	N5 vs. N1	$3.4 \cdot 10^{-24}$	$+0.136$	✓
DE	N5 vs. MQSTR	$5.5 \cdot 10^{-17}$	$+0.128$	✓
DE	N5 vs. MQSTPR	$5.1 \cdot 10^{-28}$	$+0.184$	✓
DE	N5 vs. N6	$2.0 \cdot 10^{-41}$	$+0.222$	✓
DE	N2 vs. N3	$1.5 \cdot 10^{-5}$	$+0.062$	✓
DE	N2 vs. N4	$1.6 \cdot 10^{-7}$	$+0.076$	✓
DE	MQSTR vs. MQST	$9.4 \cdot 10^{-7}$	$+0.056$	✓
DE	MQSTR vs. MQSTPR	$3.2 \cdot 10^{-7}$	$+0.056$	✓
DE	MQSTR vs. N1	1.00	$+0.008$	–
DE	MQSTPR vs. MQSTP	$1.4 \cdot 10^{-36}$	$+0.156$	✓
FR	N5 vs. N2	0.29	$+0.005$	–
FR	N5 vs. N1	$1.6 \cdot 10^{-31}$	$+0.127$	✓
FR	N5 vs. MQSTR	$2.5 \cdot 10^{-15}$	$+0.099$	✓
FR	N5 vs. MQSTPR	$2.6 \cdot 10^{-36}$	$+0.164$	✓
FR	N5 vs. N6	$4.6 \cdot 10^{-42}$	$+0.179$	✓
FR	N2 vs. N3	0.065	$+0.028$	–
FR	N2 vs. N4	$1.9 \cdot 10^{-6}$	$+0.059$	✓
FR	MQSTR vs. MQST	$3.4 \cdot 10^{-6}$	$+0.040$	✓
FR	MQSTR vs. MQSTPR	$1.2 \cdot 10^{-14}$	$+0.065$	✓
FR	MQSTR vs. N1	0.12	$+0.028$	–
FR	MQSTPR vs. MQSTP	$3.5 \cdot 10^{-55}$	$+0.174$	✓

we refit each track with the nested mixed model $\mathrm{mrr5} \sim \textsc{pipeline} * \textsc{language} + (1 \mid \textsc{topic})$, a random intercept per topic (topic nested in language). The topic random effect absorbs the dominant share of the variance, the intra-class correlation is 0.78 in the sparse track ($\sigma^2_{\text{topic}} = 0.150$ vs. $\sigma^2_{\text{resid}} = 0.041$) and 0.72 in the neural track (0.152 vs. 0.059), which quantifies why per-topic difficulty, not the system, is the largest source of variation, and why our primary per-system inference is the topic-blocked Friedman test. The pipeline×language interaction remains highly significant once topic is modelled explicitly (likelihood-ratio test against the additive model: sparse $\chi^2_{14} = 7.8 \cdot 10^3$, neural $\chi^2_{10} = 582$, both $p < 10^{-100}$), confirming that the interaction effect of Table 6 is not an artefact of treating queries as replicates.

6.4. Summary of Practical Takeaways

Table 8 distils the significance tests above into the handful of design decisions that actually matter, each stated as a before/after comparison with its sign, its typical effect size across languages, and whether the difference is statistically significant.

Table 6
Two-way ANOVA with Type-II SS on per-query MRR@5. All terms are highly significant in both tracks. The $\mathrm{pipeline} \times \mathrm{language}$ interaction is very large in the sparse track, consistent with the much larger gain that translation delivers on DE/FR than on EN.

Track	Factor	SS	F	p
sparse	pipeline	307.5	230.4	$< 10^{-300}$
sparse	language	625.6	1640.7	$< 10^{-300}$
sparse	pipeline × language	341.1	127.8	$< 10^{-300}$
neural	pipeline	58.8	55.9	$4.2 \cdot 10^{-58}$
neural	language	39.2	93.1	$4.3 \cdot 10^{-41}$
neural	pipeline × language	34.6	16.4	$4.4 \cdot 10^{-30}$

Table 7
Marginal Tukey HSD on the PIPELINE factor (pooled over the three languages, $\alpha = 0.05$). "Marg. MRR@5" is the query-weighted marginal mean; configurations that do *not* share a group letter are significantly different. The pool is dominated by EN ($\approx 75\%$ of queries), so the ranking of the lower sparse block is EN-driven (see text).

Track	Pipeline	Marg. MRR@5	Group
sparse	MQSTR	0.511	a
sparse	MQST	0.472	b
sparse	MQSTPR	0.453	b
sparse	MQS	0.384	c
sparse	MQ	0.383	c
sparse	M	0.354	d
sparse	base	0.344	d
sparse	MQSTP	0.291	e
neural	N5	0.578	a
neural	N2	0.577	a
neural	N3	0.552	b
neural	N4	0.531	c
neural	N6	0.513	c
neural	N1	0.481	d

6.5. Limitations

- **Distributional assumptions.** MRR@5 is bounded in $[0, 1]$ and concentrates almost all mass at 0 and 1 (Figure 1), so the two-way ANOVA is only approximate; the distribution-free Friedman + Wilcoxon procedure, on which we base every categorical claim, is unaffected.
- **Different query sets across languages.** Because the three languages share only partially overlapping query identifiers, the LANGUAGE main effect reflects corpus-level difficulty rather than within-query cross-lingual variation, and we do not interpret it.
- **Topics as replicates.** The two-way ANOVA treats per-query scores as independent replicates, leaving the (large) topic variance in the residual and making the PIPELINE and interaction F-tests conservative. The topic-aware nested model (Section 6.3) and the topic-blocked Friedman test (Section 6.2) respect the repeated-measures structure and agree with the ANOVA on every qualitative conclusion.

7. Conclusions and Future Work

In this paper, we presented an effective information retrieval pipeline designed to match multilingual social-media claims with their corresponding scientific publications, together with a parallel exploration of fully neural alternatives executed on the departmental GPU cluster. Driven by the observation, drawn

Table 8
Practical takeaways from the significance tests of this section. "Effect" is the typical mean-MRR@5 change across the three languages; "Sig." reports whether the paired Wilcoxon (Holm-corrected) difference is significant at $\alpha = 0.05$ on the languages indicated.

Design decision	Effect	Sig.	Takeaway
Domain fine-tuning (**N1**→**N2**)	$+0.09$ to $+0.14$	yes (all)	The single decisive lever; specializing the dense retriever dominates.
doc2query (**N2**→**N5**)	$+0.00$ to $+0.01$	no	Small, cheap, consistent, but not statistically distinguishable from **N2**.
Cross-encoder re-rank (**N2**→**N3**/**N4**)	-0.02 to -0.08	yes (mostly)	Hurts on top of a specialized bi-encoder, even when task fine-tuned.
RRF + re-rank (**N5**→**N6**)	-0.02 to -0.22	yes (all)	Stacking fusion and re-ranking is the largest regression; collapses on DE.
Translation (**MQS**→**MQST**, sparse)	$+0.27$/$+0.39$ on DE/FR	yes (DE/FR)	Essential for cross-lingual recall; a no-op on EN.
PRF (**MQST**→**MQSTP**, sparse)	≈ -0.18	yes (all)	Consistently harmful; drops below the BM25 baseline.
Re-rank on clean candidates (**MQST**→**MQSTR**)	$+0.04$ to $+0.06$	yes (all)	Re-ranking helps only when fed a PRF-free candidate set.
Best neural vs best sparse (**N5** vs **MQSTR**)	$+0.05$ to $+0.13$	yes (all)	The submitted neural run beats the strongest sparse pipeline everywhere.

from the working notes of previous editions, that competitive systems on this task tend to rely on dense retrieval and learned re-ranking [2], we deliberately developed the two tracks side by side and used them to inform each other.

The sparse track, starting from a basic single-field BM25 implementation [4] that suffered from severe vocabulary mismatch, was progressively enhanced through targeted query pre-processing, synonym compression, MarianMT-based multilingual translation, multi-field weighted search and TinyBERT cross-encoder [5] re-ranking (with pseudo-relevance feedback [9] explored but ultimately discarded due to query drift), reaching a peak MRR@5 of 0.525 on English, 0.414 on German, and 0.510 on French.

The neural track, in turn, made it clear that the dominant lever on this collection is *domain specialization* of the dense retriever. The largest single improvement was produced not by any re-ranking stage but by contrastive fine-tuning of the bi-encoder [5] on the (post, gold paper) pairs of the training splits. A second, smaller but consistent gain came from doc2query expansion [13] of the abstracts, which rewrites each document with synthetic, claim-like queries and bridges the stylistic gap between informal posts and academic prose. This configuration, doc2query plus fine-tuned bi-encoder (**N5** in Section 5), was the strongest of the fully evaluated variants, reaching an average MRR@5 of 0.575 on the test split, and is the run we eventually submitted to the official task. By contrast, every additional layer of complexity that we tested on top of **N5**, from a generic to a fine-tuned cross-encoder re-ranker and a Reciprocal Rank Fusion stage [17] with sparse BM25, regressed performance, showing that on this collection re-ranking on top of an already strong dense retriever is more likely to harm than to help.

The one component we explicitly designed but did not have time to evaluate end-to-end is an LLM-based listwise re-ranker [6] on top of **N5**. The expectation is that an instruction-tuned LLM, unlike a similarity-based reranker, can perform an explicit reasoning step over each candidate paper given the post, and that this kind of supervision is the most plausible way to recover the gains that cross-encoder rerankers failed to deliver in our experiments.

Future work follows directly from these findings, and converges on the neural track that upcoming

work will pursue at full scale. First, with reliable GPU access we plan to run the LLM listwise re-ranking variant of the pipeline end-to-end on the development and test sets, and to fully automate the tuning of its hyperparameters, including the interpolation weight α that the CPU-only environment prevented us from optimizing on the sparse side. Second, to address translation inaccuracies involving highly specific scientific entities, we will explore hybrid query representations [12] that concatenate the original multilingual post with its English translation, so as to preserve exact-match named entities, gene symbols and acronyms during the translation phase. Finally, while our hybrid sparse-plus-dense fusion attempt (**N6**) regressed below the simpler **N5**, we believe a more principled fusion strategy, applied earlier in the pipeline and tuned on a held-out split, may still combine the lexical strengths of BM25 with the semantic strengths of the bi-encoder; we plan to revisit it once the LLM re-ranking stage is in place.

8. Declaration on Generative AI

During the preparation of this work, the authors used Gemini and Claude 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] J. M. Struß, S. Schellhammer, S. Dietze, V. V., V. Setty, T. Chakraborty, P. Nakov, A. Anand, P. Chungkham, S. Hafid, D. Sahnan, K. Todorov, The CLEF-2026 CheckThat! lab: Advancing multilingual fact-checking, in: R. Campos, A. Jatowt, Y. Lan, M. Aliannejadi, C. Bauer, S. MacAvaney, A. Anand, Z. Ren, S. Verberne, N. Bai, M. Mansoury (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2026, pp. 325–335.

[2] S. Schellhammer, S. Hafid, Y. S. Kartal, K. Boland, D. Dimitrov, K. Todorov, S. Dietze, Overview of the CLEF-2026 CheckThat! lab task 1 on source retrieval for scientific web claims, CLEF 2026, Jena, Germany, 2026.

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

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

[5] S. Marchesin, A. Purpura, G. Silvello, Focal elements of neural information retrieval models. an outlook through a reproducibility study, Information Processing & Management 57 (2020) 102109. URL: https://www.sciencedirect.com/science/article/pii/S0306457319301864. doi:https://doi.org/10.1016/j.ipm.2019.102109.

[6] W. Sun, L. Yan, X. Ma, S. Wang, P. Ren, Z. Chen, D. Yin, M. de Rijke, Is ChatGPT good at search? Investigating large language models as re-ranking agents, in: Proc. 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023), Association for Computational Linguistics, 2023, pp. 14918–14937.

[7] X. Zhou, R. Zafarani, A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities, ACM Computing Surveys (CSUR) 53 (2020) 109:1–109:40.

[8] A. Barrón-Cedeño, T. Elsayed, P. Nakov, G. Da San Martino, M. Hasanain, R. Suwaileh, F. Haouari, N. Babulkov, B. Hamdan, A. Nikolov, S. Shaar, Z. Sheikh Ali, Overview of CheckThat! 2020: Automatic Identification and Verification of Claims in Social Media, in: A. Arampatzis, E. Kanoulas, T. Tsikrika, S. Vrochidis, H. Joho, C. Lioma, K. Eickhoff, A. Névéol, L. Cappellato, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Eleventh International Conference of the CLEF Association (CLEF 2020), Lecture Notes in Computer Science (LNCS) 12260, Springer, Heidelberg, Germany, 2020, pp. 215–236.

[9] I. Ruthven, M. Lalmas, A survey on the use of relevance feedback for information access systems, The Knowledge Engineering Review 18 (2003) 95–145.

[10] H. Ziak, Kern, Evaluation of Pseudo Relevance Feedback Techniques for Cross Vertical Aggregated Search, in: J. Mothe, J. Savoy, J. Kamps, K. Pinel-Sauvagnat, G. J. F. Jones, E. SanJuan, L. Cappellato, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Sixth International Conference of the CLEF Association (CLEF 2015), Lecture Notes in Computer Science (LNCS) 9283, Springer, Heidelberg, Germany, 2015, pp. 91–102.
[11] D. A. Hull, Stemming Algorithms: A Case Study for Detailed Evaluation, Journal of the American Society for Information Science (JASIS) 47 (1996) 70–84.
[12] J.-Y. Nie, Cross-Language Information Retrieval, Morgan & Claypool Publishers, USA, 2010.
[13] R. Nogueira, K. Cho, Document expansion by query prediction, arXiv preprint arXiv:1904.08375 (2019).
[14] J. Wang, D. W. Oard, Combining Bidirectional Translation and Synonymy for Cross-Language Information Retrieval, in: E. N. Efthimiadis, S. Dumais, D. Hawking, K. Järvelin (Eds.), Proc. 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), ACM Press, New York, USA, 2006, pp. 202–209.
[15] M. Sanderson, Test Collection Based Evaluation of Information Retrieval Systems, Foundations and Trends in Information Retrieval (FnTIR) 4 (2010) 247–375.
[16] D. K. Harman, Information Retrieval Evaluation, Morgan & Claypool Publishers, USA, 2011.
[17] S. Wu, Data Fusion in Information Retrieval, Springer-Verlag, Heidelberg, Germany, 2012.
[18] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A Next-generation Hyperparameter Optimization Framework, in: Proc. 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2019), ACM Press, 2019, pp. 2623–2631.
[19] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30. URL: https://jmlr.org/papers/v7/demsar06a.html.