NEXA at CheckThat! 2026: Hybrid Source Retrieval for Cross-Lingual Scientific Web Claims

Notebook for the CheckThat! Lab at CLEF 2026

Paul Arlot[1], Andrea Di Tillo[1], Gaute Greiff Flagstad[1], Bita Khashechian[1], Danil Smirnov[1], Marco Tomaiuoli[1] and Nicola Ferro[1]

[1]*University of Padua, Italy*

Abstract

This paper presents the system developed by Team NEXA for Task 1 of the CLEF 2026 CheckThat! Lab, focusing on the retrieval of scientific sources for claims found on social media. Given the challenge of linking informal web discourse to formal scientific literature across English, German, and French, we propose a multi-stage retrieval framework. Our approach combines traditional lexical search using BM25 with neural dense retrieval based on multilingual transformer embeddings, using score fusion to integrate lexical and semantic signals. We also investigate cross-encoder re-ranking as an additional refinement step. The system aims to bridge the linguistic and stylistic gap between social media posts and academic abstracts, providing a scalable solution to assist fact-checkers in verifying scientific information.

Keywords

Information Retrieval, Multilingual Retrieval, Hybrid Retrieval, Dense Retrieval, BM25, BGE-M3, Score Fusion, Cross-Encoder Re-ranking, Neural Re-ranking, Lucene, CLEF CheckThat!

1. Introduction

Social media is full of scientific claims, but they are usually oversimplified, informal, and stripped of context. For fact-checkers, tracking down the original study behind a tweet or post is a painfully slow task—especially when the post paraphrases the study or uses a different language. Task 1 of the CLEF 2026 CheckThat! Lab [1] tackles this issue. The goal is to automatically link social media claims to their scientific sources. This comes with two major hurdles: a massive style gap between casual internet slang and formal abstracts, and a cross-lingual challenge (matching English, French, and German posts against a mostly English corpus). We address this with a highly configurable IR system. First, we translate all non-English text into English to keep our pipeline consistent. The retrieval strategy pairs traditional BM25 search (utilizing structured fields like title, abstract, and authors) with dense multilingual embeddings to capture semantic similarity. Our system supports a wide range of preprocessing and indexing configurations, allowing us to experiment with different setups to bridge the vocabulary gap. Finally, we evaluate the impact of query expansion and cross-encoder reranking on overall retrieval performance.

This paper is organized into five sections. Section 2 reviews some related work; Section 3 presents our approach in detail; Section 4 describes the experimental setup; Section 5 discusses our main findings and analyzes the obtained results; and Section 6 concludes the paper and describes possible future work.

CLEF 2026 Working Notes, 21 – 24 September 2026, Jena, Germany

paullouisjean.arlot@studenti.unipd.it (P. Arlot); andrea.ditillo@studenti.unipd.it (A. Di Tillo); gaute.greiff.flaegstad@studenti.unipd.it (G. G. Flagstad); bita.khashechian@studenti.unipd.it (B. Khashechian); danil.smirnov@studenti.unipd.it (D. Smirnov); marco.tomaiuoli@studenti.unipd.it (M. Tomaiuoli); nicola.ferro@unipd.it (N. Ferro)

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

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

2. Related Work

2.1. Scientific Claim Verification and the CheckThat! Lab

Automatic fact-checking has been an active research area since the emergence of computational approaches to identifying misinformation on the web. The CLEF CheckThat! Lab [2] organises a series of shared tasks focused on the full fact-checking pipeline, from detecting check-worthy claims to verifying them against evidence. Early editions [3, 4, 5] addressed claim worthiness detection and tweet-level verification in Arabic and English. Task 1 of CheckThat! 2026 closely resembles Subtask 4b of CheckThat! 2025 [6], with the primary distinction being the introduction of multilingual datasets. The 2026 edition requires, given a social media post that paraphrases a scientific finding, identifying the original publication in a multilingual corpus spanning English, French, and German. This framing places the task squarely in the domain of ad-hoc retrieval [7], where each claim acts as a query and each paper abstract is a candidate document, but it adds the complication of a large stylistic gap between informal claim language and formal scientific writing.

Among prior approaches to the 2025 edition, the Deep Retrieval system [8] is of particular relevance due to its successful integration of both dense and sparse retrieval techniques, an approach we also adopt and extend with cross-encoder reranking.

2.2. Lexical Retrieval

The BM25 ranking function [9] remains a strong and widely used baseline for ad-hoc retrieval despite being over three decades old. It extends classical TF-IDF weighting with a saturation function on term frequency and a document-length normalisation parameter, giving it robustness to document length variation. Implementations in Apache Lucene [10] are highly optimised and support fielded queries, allowing different weights to be assigned to title, abstract, and author fields. For tasks involving formal scientific text, BM25 already performs well because the vocabulary of abstracts is precise and largely non-ambiguous; the main weakness is the vocabulary gap between informal claim language and the scientific abstracts it must match against.

Query expansion has been studied as a remedy for vocabulary mismatch. Relevance feedback, introduced by Rocchio [11], reformulates queries by adding high-weight terms from the top-ranked documents retrieved by a first pass. In the pseudo-relevance feedback (PRF) variant the system assumes the top k results are relevant without manual judgement. While effective in many settings, PRF is sensitive to topic drift: in a corpus where most documents cluster around the same scientific domain (here, COVID-19 and related health topics), the re-expanded query moves toward the dominant cluster rather than the specific claim. External lexical resources such as WordNet provide another expansion route by injecting synonyms, but they cover general-vocabulary words rather than scientific terminology, which limits their usefulness in this task.

2.3. Dense and Hybrid Retrieval

Transformer-based bi-encoders have shifted the state of the art in retrieval by encoding queries and documents independently into a shared dense space and using cosine similarity for ranking [7]. Unlike BM25, bi-encoders capture semantic similarity rather than lexical overlap, which is precisely what is needed to bridge informal paraphrases and formal abstracts. BGE-M3 [12] is a multilingual dense encoder that extends this paradigm to more than 100 languages in a single model, covering all three languages of the CheckThat! corpus without requiring separate encoders or translation at inference time. Its training mixture includes informal-to-formal cross-domain text pairs, which makes it a natural fit for the claim-to-abstract matching problem we address.

Hybrid retrieval combines lexical and dense signals to benefit from the complementary strengths of both. BM25 is precise for queries that contain exact vocabulary from the target document; dense retrieval captures paraphrases and domain-adjacent concepts. Score fusion, where normalised BM25 and cosine scores are combined via a weighted sum, is a standard approach that consistently outperforms

either signal alone [7]. In our system we adopt min-max normalisation within each query's candidate set before combining, so the fusion weight α has a stable interpretation across queries.

SPECTER2 [13], a domain-specific encoder trained on scientific paper citation proximity, represents a different hypothesis: that scientific specialisation outweighs multilingual coverage for a scientific corpus. The model is adapted from a citation-graph training objective to an ad-hoc query adapter for scientific retrieval tasks. Our experiments show that this hypothesis does not hold for the CheckThat! task (Section 5.6), because the query-side representations of tweet-style claims and the document-side representations of abstracts are geometrically incompatible in SPECTER2's embedding space, since neither informal social media text nor cross-lingual bridging were part of its training distribution.

2.4. Cross-Encoder Reranking

Two-stage retrieval pipelines, where a fast first-stage retriever generates a short candidate list and a slower but more expressive second-stage model reranks it, are standard in modern IR [7]. Cross-encoders process each (query, document) pair jointly rather than encoding them independently, which allows them to model fine-grained token-level interactions at the cost of being too slow for full-collection retrieval. Pre-trained on large passage retrieval datasets such as MS MARCO, the `ms-marco-MiniLM-L-6-v2` family delivers strong reranking quality at a fraction of the inference time of larger models.

Fine-tuning cross-encoders on task-specific hard negatives has been shown to improve performance on downstream retrieval tasks by teaching the model to discriminate between genuinely relevant and near-miss candidates. In our pipeline, hard negatives are sampled from the top-k lexical retrieval results that did not match the gold publication, which are harder and more informative than random collection samples. The asymmetric outcome of fine-tuning across languages — gains in French and German but degradation in English — is consistent with observations in multilingual reranking literature, where the base checkpoint calibration for the dominant training language (English) is difficult to shift with a small task-specific dataset.

2.5. Multilingual Retrieval

Multilingual information retrieval requires resolving both the vocabulary mismatch between query and document and the language mismatch when they are written in different languages. Translate-then-retrieve pipelines apply machine translation to either the queries, the documents, or both before running a monolingual retrieval system. While this adds translation latency and introduces translation errors, it allows a well-tuned monolingual system (such as a BM25 ranker optimised on English text) to be applied uniformly across languages. In our system we translate all non-English claims into English at query time and index all documents in English, so that a single analyzer and retrieval configuration applies to all three languages. BGE-M3, by contrast, operates natively on the original languages without translation, demonstrating stronger multilingual bridging at the dense stage and reducing the system's dependence on translation quality.

3. Methodology

In this section, we describe the architecture and components of our system. Our approach follows a multi-stage Information Retrieval pipeline combining traditional lexical matching with semantic representations.

3.1. Data Parsing

The first stage of the pipeline turns the two textual inputs of the system, the corpus of scientific publications and the set of input claims, into structured Java objects that the rest of the pipeline can work with. Both inputs are provided as JSON and share the same text cleaning routine, so we describe

them together. On the data side we have two simple classes: `Publication` for documents and `Claim` for queries.

3.1.1. Documents (Publications)

Documents are read through an abstract `CommonParser` class that defines the iterator interface and exposes the shared text cleaning routine, with concrete subclasses for each input format. Since the corpus is provided as JSON, the only subclass we currently use is `JsonParser`.

`CommonParser` implements `Iterator<Publication>`, so it produces one `Publication` at a time as the input is read. Each `Publication` gets the fields we use for retrieval: a numerical identifier (`pubkey`), the `title`, `abstract`, `venue` and `authors`. Missing or `null` fields are replaced with empty defaults, and empty abstracts are filled with a placeholder token so that every document still produces a valid entry in the index.

`JsonParser` uses the Jackson library to read the corpus, which is stored as a JSON array of document objects. We process the stream incrementally rather than loading the whole file at once, which keeps memory usage flat regardless of corpus size. For each publication we map the JSON fields onto a `Publication` instance and run the abstract through the cleaning routine before it leaves the parser.

3.1.2. Queries (Claims)

Queries are loaded from a separate JSON topics file containing a list of claim objects. We parse it into a `List<Claim>` using a Jackson `ObjectMapper`, rather than going through the streaming parser used for documents. Each `Claim` keeps all three fields: the claim `index`, the claim `text`, and the gold `pubkey` when present (used only at evaluation time).

To keep documents and queries aligned, the cleaning routine described next is applied to the claim text from inside the setter of the `Claim` class. By the time the searcher iterates over the list, both sides of the retrieval problem have gone through exactly the same preprocessing, which avoids the kind of subtle mismatches that arise when documents and queries are normalized differently.

3.1.3. Text Cleaning

Before any text reaches the analyzer, we pass it through a cleaning function (`cleanText`) that strips out the noise produced by the mix of scientific writing and social media content in the corpus. The function applies the following steps in order:

- **Unicode normalisation:** the text is converted to NFC form so the character representation is consistent.
- **HTML unescaping and stripping:** HTML entities are decoded and any leftover tags are removed.
- **URL removal:** hyperlinks are dropped so they do not end up as tokens.
- **Citation removal:** bracketed references (e.g. "[1, 2–5]") and author–year citations (e.g. "(Rossi et al., 2023)") are removed.
- **Section header removal:** structural labels that appear in scientific abstracts, such as *Abstract*, *Methods*, *Results* or *Conclusions*, are stripped.
- **Symbol and emoji removal:** mathematical and scientific symbols (e.g. $^\circ$, $\pm$, $\geq$) and emoji characters are replaced with spaces.
- **Social media artefacts:** Twitter-style mentions (e.g. `@user`) are removed.
- **Whitespace collapsing:** runs of whitespace are collapsed into a single space and any leading or trailing whitespace is trimmed.

This step is important because the corpus mixes formal abstracts with informal claims. Without it the analyzer would have to deal with a lot of noisy tokens that could decrease matching quality.

3.2. Query Expansion

To merge the vocabulary between informal social media claims and formal scientific abstracts, the system supports four query expansion strategies that can be turned on independently from the configuration file. Each one adds extra terms to the Boolean query produced by the searcher, but with different sources and different boost weights so that the original claim terms always weigh more than the expansions.

3.2.1. Static Dictionary Expansion

The simplest strategy is to use tested dictionaries for all the three languages, loaded from the file referenced by the `synonymsFile` parameter. During query construction (`buildLexicalQuery`), each token of the claim is looked up in the dictionary, and any matching synonyms (for example, "car" $\rightarrow$ "automobile", "vehicle") are added to the abstract subquery with a boost of `0.5f`, half the weight of the original term. This keeps the original tokens dominant in the ranking while still letting the expansion contribute. Retrieved from: "https://www.openthesaurus".

3.2.2. LLM-based Expansion

To capture relations beyond a fixed dictionary, the system can expand the query using a Large Language Model. The claim text is sent to the Groq API with the configured `openApiKey` and `llmModel` (currently `llama-3.1-8b-instant`). The model returns a short list of related terms, for example "CO_2", "pollution" and "glaciers" for "climate change". These terms are added to the title and abstract subqueries with a boost of `0.4f`.

This route is more expressive than the static dictionary, but it adds a network round trip to every query, and the free tier of the Groq API is rate-limited to roughly 30 requests per minute, which makes it impractical for batch experiments without throttling.

3.2.3. Pseudo-Relevance Feedback

The third strategy is Pseudo-Relevance Feedback (PRF), which expands the query using the local index instead of an external resource. PRF works on the assumption that the top-ranked documents returned by a first search are likely to be relevant, and reuses their content as a source of related terms. The procedure has two stages:

1. **Initial search.** The system runs the original query and keeps the top-k documents (we use $k = 3$).
2. **Expansion and re-search.** The title and abstract of each document are tokenised, and standard stopwords as well as numeric or excessively short tokens are removed. From the remaining terms, the five most frequent are appended to the original query with a boost of `0.3f`. Finally, the expanded search is re-executed against the index.

The primary appeal of Pseudo-Relevance Feedback (PRF) lies in its data-driven nature: since expansion terms are extracted directly from the top-retrieved documents of the corpus, they are inherently guaranteed to be in-vocabulary for the index. This effectively mitigates the vocabulary mismatch problem without relying on external resources. However, this effectiveness comes at a significant computational cost. Under this framework, processing each claim requires two sequential Lucene searches instead of one—the first to retrieve the feedback documents and the second to execute the expanded query—which roughly doubles the overall retrieval time and increases system latency. Consequently, due to these efficiency constraints, PRF was disabled during the timed runs reported in our efficiency evaluation.

3.2.4. Earlier Gemini-based Expansion

An earlier version of the LLM expansion module relied on Google's Gemini API rather than Groq, with a different integration shape. A small Python service ran alongside the Java search engine and acted as a bridge between the two. For each query, the service first looked up the claim text in a local on-disk cache; if the same query had been expanded before, the cached terms were returned immediately. Otherwise the raw claim was forwarded to the Gemini API with a prompt asking the model to ignore emojis and opinions, list related synonyms, and return between 10 and 15 keywords. The response was then written back to the cache and returned to the Java side.

This approach was not carried into the final pipeline. It required running a separate Python service alongside the Java search engine, with its own dependency stack, cache file, and HTTP coupling point between the two processes. The Groq-based path described above covers the same use case (LLM-generated synonyms with on-disk caching) directly from Java, with one fewer moving part. The Gemini API was also less reliable in our setup, which made the trade-off easier.

3.3. Text Analysis

The text analysis phase transforms raw textual input into a stream of tokens that can be indexed and later retrieved. Initially, we tried to use language-specific analyzer classes but using a single English analyzer after translation was simpler and offered better results. The analysis is performed through an analyzer class, which extends the Apache Lucene `Analyzer` framework.

The analyzer follows an architecture composed of three main components:

- **Tokenizer:** Responsible for splitting the input text into raw tokens.
- **Token Filters:** Sequentially modify, normalize, or enrich the token stream.
- **Stemming/Lemmatization:** Reduces tokens to their base or root forms to improve generalization.

The overall pipeline is dynamically configured through external YAML configuration files, allowing flexible adaptation of processing strategies and experimental settings.

3.3.1. Tokenizer Configurations

Multiple tokenization strategies are supported, enabling the system to handle different types of input text:

- **Standard Tokenizer:** Uses Lucene's rule-based tokenizer to handle punctuation, digits, and word boundaries.
- **Letter Tokenizer:** Emits tokens composed only of alphabetic characters, treating non-letter characters as delimiters.
- **Whitespace Tokenizer:** Splits input strictly on whitespace characters without processing punctuation.
- **NLP Tokenizer:** Uses Apache OpenNLP models to perform linguistically informed tokenization.

The tokenizer type is selected through configuration files, allowing to try different strategies.

3.3.2. Token Filters

Token filters are applied after tokenization to normalize and enrich the token stream. These filters are categorized into general (language-independent) and language-specific filters.

General Token Filters These filters perform general normalization operations across all languages:

- **LowerCaseFilter:** Converts all tokens to lowercase.
- **ICUFoldingFilter:** Normalizes accented and special characters into ASCII equivalents.
- **NBSPFilter:** Removes non-breaking spaces and trims tokens.
- **RemoveDuplicatesTokenFilter:** Eliminates duplicate tokens.
- **LengthFilter:** Removes tokens that are too short or too long.
- **RepeatedLetterFilter:** Normalizes tokens with repeated characters (e.g., "soooo" → "soo").

Enrichment Filters To improve semantic representation, additional filters are applied:

- **AbbreviationExpansionFilter:** Expands abbreviations using predefined mappings (e.g., "ml" → "machine learning"). Retrived from: https://wordnet.princeton.edu/
- **CompoundPOSTokenFilter:** Uses Part-of-Speech tagging to combine semantically related tokens into compound terms.
- **ShingleFilter:** Generates n-grams to capture local contextual information.
- **PositionFilter:** Normalizes positional increments for indexing consistency.

Language-Specific Filters Each analyzer applies additional filters tailored to linguistic characteristics:

- English-specific filters (e.g., possessive removal, English stopwords)
- French-specific filters (e.g., elision handling, French stopwords)
- German-specific filters (e.g., compound handling and normalization)

This design ensures that each language is processed according to its grammatical and lexical properties. Since we translate everything to English, we are using only English-specific filters.

3.3.3. Stemming and Lemmatization

The system supports multiple strategies for reducing tokens to their canonical forms: **Porter Stemmer**, **Snowball Stemmer**, **OpenNLP Lemmatizer**, **No Stemming**.

The selected method is controlled through configuration files and can vary depending on the experimental setup.

3.4. Document Indexing

The document indexing stage transforms processed publications into a searchable inverted index using Apache Lucene. This structure enables efficient retrieval of relevant documents during the query phase. In our system, indexing is handled by the `DirectoryIndexer` component, which coordinates parsing, preprocessing, and storage of documents.

3.4.1. Indexing Workflow

The indexing workflow operates over a collection of JSON files containing scientific publications. Each file is parsed to extract individual documents, represented internally as structured `Publication` objects. Only documents containing meaningful textual content (such as title or abstract) are considered for indexing.

In addition to standard preprocessing, the system supports optional enhancements. When enabled, non-English documents are translated into English before indexing using a locally hosted Gemma model served via a dedicated Python HTTP service; the same service is applied to non-English claims at query time. Moreover, the system can generate dense vector representations of the textual content

by interacting with an external embedding service. These embeddings are stored alongside the indexed documents to support semantic retrieval.

Each processed publication is converted into a Lucene `Document` and added to the index using an `IndexWriter`. During this process, statistics such as the number of indexed documents and total indexing time are recorded.

3.4.2. Field Representation

Document content is stored in structured fields, such as title and abstract. A custom field type is defined to support advanced indexing features, including term frequencies, positional information, and character offsets.

The system also enables storage of term vectors, which allows more advanced retrieval functionalities such as phrase matching, proximity queries, and potential re-ranking strategies. Additionally, the original textual content can be stored in the index to support downstream processing steps.

3.4.3. Processing Pipeline

The indexing procedure follows a well-defined sequence of steps. First, the system initializes the index writer with the configured analyzer and output directory. Then, for each document, the following operations are performed:

- Parsing the input file into structured publication objects
- Filtering out incomplete or invalid entries
- Detecting the language of the document
- Applying optional translation if required
- Generating semantic embeddings (if enabled)
- Converting the document into a Lucene-compatible format
- Adding the document to the index

Once all documents have been processed, the index is finalized by committing changes and closing the writer.

This modular and extensible design allows the indexing system to integrate multiple preprocessing strategies, multilingual handling, and semantic enrichment, while keeping the core indexing logic independent and reusable.

3.5. Retrieval Phase

The retrieval phase is responsible for matching input claims with relevant scientific publications stored in the indexed collection. This component is implemented through the `Searcher` class, which performs query processing, executes search operations over the Lucene index, and outputs ranked results.

The system initializes a Lucene `IndexSearcher` over the indexed data and employs the BM25 similarity function for ranking. Queries are provided as a collection of claims stored in JSON format, where each claim represents a short textual statement that must be linked to relevant documents.

3.5.1. Query Formulation

Each claim is processed independently. The textual content is first transformed into a structured query using a `QueryBuilder`, which applies the same analyzer used during indexing. This ensures that both documents and queries undergo consistent preprocessing, reducing mismatches due to tokenization or normalization.

The resulting terms are used to construct field-specific queries targeting different parts of the indexed documents.

3.5.2. Structured Boolean Queries

Instead of relying on a single-field representation, the system generates separate subqueries for the title and abstract fields of each document.

These subqueries are combined using a Boolean query with OR clauses, allowing documents to be retrieved if they match either field. This formulation increases recall while maintaining flexibility in matching.

To further refine the ranking, different importance weights are assigned to each field through boosting. In particular, matches occurring in the abstract field are given higher importance than those in the title, since social media claims typically paraphrase scientific findings rather than paper titles. This results in a weighted query structure that prioritizes the abstract as the primary evidence field while still drawing a smaller signal from the title.

3.5.3. Ranking Mechanism

Once the Boolean query is constructed, it is executed using the Lucene search engine. The system retrieves the top-k documents according to their BM25 scores. This ranking function considers the frequency of query terms, their rarity across the collection, and document length normalization.

The number of retrieved documents is configurable and can be adjusted depending on the evaluation setup.

3.5.4. Result Formatting

The retrieved results are written to an output file following the TREC evaluation format. Each entry includes the query identifier, document identifier, rank position, score, and run identifier. This standardized format ensures compatibility with external evaluation tools commonly used in information retrieval benchmarks.

3.6. Cross-Encoder Fine-Tuning

After the initial retrieval stage, the system optionally applies a cross-encoder reranker that scores each (claim, candidate) pair jointly. Unlike the bi-encoder used for dense fusion, a cross-encoder receives both texts concatenated and produces a single relevance score, which allows it to model fine-grained interactions between the claim and the abstract but makes it too slow for full-collection retrieval. It is therefore applied only to the short candidate list produced by the earlier stages.

We used the `ms-marco-MiniLM-L-6-v2` checkpoint as our base model and fine-tuned it on task-specific data to adapt it to the informal-claim-to-scientific-abstract matching task. To ensure the reproducibility of our experiments, all data shuffling, negative sampling, and weight initialisations were locked using a fixed seed of 15. The training set was constructed by pairing each claim in the training split with its corresponding gold publication as the positive example. Hard negatives were then sampled from the top-k candidates returned by the lexical retrieval stage, excluding those matching the gold pubkey. These candidates represent documents that the system falsely ranked highly, providing a stronger and more informative training signal than random negatives drawn from the entire collection. The model was fine-tuned for 2 epochs with a maximum sequence length of 512 tokens, with the training loss converging to approximately 0.22. The impact of this fine-tuning process on downstream retrieval performance is reported in Section 5.

4. Experimental Setup

4.1. Used Collections

We used the dataset released for CheckThat! 2026 Task 1, Source Retrieval for Scientific Web Claims [14]. The dataset consists of a scientific publication collection and multilingual social media claim datasets in

English, French, and German. The publication corpus is provided in `collection_data.json`. The claim data is divided into training, development, and final test sets for each language. We used the development data only for hyperparameter tuning, and all scores reported in Section 5 were computed on it. The final test data was used only for the official submission and was not used for hyperparameter tuning.

4.2. Evaluation Measure

The main evaluation measure used is the Mean Reciprocal Rank at 5 (MRR@5):

$$MRR@5 = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \begin{cases} \frac{1}{rank_i} & \text{if } rank_i \leq 5 \\ 0 & \text{if } rank_i > 5 \end{cases}$$

With:

- $|Q|$: The total number of queries being evaluated.
- i: The index of the current query (from 1 to $|Q|$).
- $rank_i$: The rank position of the *first* relevant result for query i.
  - If the first relevant result is at rank 1, 2, 3, 4, or 5, its score is $\frac{1}{rank_i}$.
  - If the first relevant result is at rank 6 or below (or if there are no relevant results), its score is 0.

4.3. Git Repository

The entire project, including the source code and selected experimental runs, is available on Bitbucket at https://bitbucket.org/upd-dei-stud-prj/seupd2526-nexa/. The repository contains the source code of the system, the initial report produced at the end of the evaluation phase, and selected runs in `trec_eval` format.

4.4. Software Environment

The system was implemented using the following software components:

- **Java / retrieval pipeline:** Java 21, Apache Maven 3, and Apache Lucene 10.4.0.
- **Python / ML services:** Python 3.9, PyTorch, Hugging Face Transformers, Sentence-Transformers, FastAPI, and Uvicorn.
- **Multilingual processing:** EasyNMT, fastText, and Google Generative AI SDK.
- **Evaluation and analysis:** `trec_eval`, NumPy, SciPy, and Matplotlib.

4.5. Hardware Environment

All experiments were executed on a machine with the following specifications:

- CPU: M4 (6 efficiency cores and 4 performance cores)
- RAM: 16GB
- GPU: 10-core
- Storage: 512GB SSD

5. Results and Discussion

In this section we report the experiments that led to the final lexical and hybrid configurations of the system. Because the search space is much larger than what we could grid-search, we tuned the components in order, fixing the winner of each phase before moving on:

1. Section 5.1 fixes the analyzer pipeline (tokenizer, stemmer, stop list, filters).
2. Section 5.2 fixes the retrieval model and BM25 parameters.
3. Section 5.3 fixes the fielded query (boosts, query mode, IDF pruning).
4. Section 5.4 reports two expansion strategies (WordNet, PRF) that did not work.
5. Section 5.5 ablates the components of the final lexical system.
6. Section 5.6 reports the dense fusion and reranking results.
7. Section 5.7 compares the runs we submitted.

All scores are MRR@5 on the dev split, reported per language (EN, FR, DE) with the average across languages. Hybrid and reranking experiments operate on the top-50 candidates of the lexical run. The dataset, dev splits, hardware, and evaluation tooling are described in Section 4.

5.1. Lexical Analyzer Tuning

We start by fixing the analyzer pipeline that the rest of the chapter builds on. Table 1 varies the tokenizer (StandardTokenizer vs. WhitespaceTokenizer) and the stemmer (KStem, Snowball, Porter, EnglishMinimal, or none), with the English stop list either active or disabled. All other components (lowercasing, ICU folding, the EnglishPossessiveFilter) are held constant, so each row isolates one change. Word n-grams are reported separately below.

Table 1
Tokenizer and stemmer comparison on the EN/FR/DE dev sets, MRR@5. Variants share the standard pipeline (lowercase, ICU folding, English stop list) and differ only in the stemmer or tokenizer.

Configuration	EN	FR	DE	Avg
Standard, no stop, no stem	0.4826	0.4453	0.3014	0.4098
Standard, stop, no stem	0.4864	0.4674	0.3225	0.4254
Standard, stop, EnglishMinimal	0.4894	0.4817	0.3304	0.4338
Standard, stop, Snowball	0.4869	0.4785	0.3358	0.4337
Standard, stop, Porter	0.4867	0.4801	0.3381	0.4350
Standard, stop, KStem	**0.4883**	**0.4901**	**0.3354**	**0.4379**
Whitespace, stop, KStem	0.4422	0.4087	0.2746	0.3752

The tokenizer dominates every other choice. Switching from the StandardTokenizer to the WhitespaceTokenizer costs about six points of average MRR@5 (0.4379 vs. 0.3752) in all three languages, because the WhitespaceTokenizer keeps punctuation attached to tokens and does not split hyphenated compounds such as “COVID-19” or “SARS-CoV-2”. The Standard pipeline also handles FR and DE without language-specific rules; we tried language-specific synonym and POS filters on those two languages and disabled them, since they added noise without measurable gain.

Within the Standard rows, the stop list matters more than the choice of stemmer. Removing it drops the average by about 1.6, and by 2.3 on FR, where common function words match aggressively against translated abstracts. The four stemmers, by contrast, fall within 0.005 of each other on the average. KStem is narrowly best and the most conservative of the four: it conflates *vaccination*/*vaccinate* but leaves *corona*/*coronary* distinct, which matters in a domain where over-stemming can reduce retrieval accuracy. We therefore lock in the Standard tokenizer, the English stop list, and KStem for the rest of the chapter.

We also tested word n-grams on top of this pipeline. Two-grams averaged 0.3855 and three-grams 0.4040, both well below the 0.4379 unigram baseline. Indexing unigrams plus bigrams reduces BM25's IDF weight on the precise scientific terms in the original sentence, and since claims are already complete sentences, n-grams add no phrase signal that unigram BM25 was missing. n-grams are disabled in all later experiments. The EnglishPossessiveFilter, which strips trailing *'s*, changes the average by less than 0.001, so we keep it on at no runtime cost.

5.2. Retrieval Model and BM25 Parameters

With the analyzer fixed, we select and optimize a retrieval ranking function. Table 2 evaluates BM25 against competing approaches: the Classic TF-IDF model, LM Dirichlet (with $\mu \in \{500, 1000\}$), and LM Jelinek-Mercer (with λ=0.5). We then systematically test BM25's hyperparameters (k_1 and b) to verify that Lucene's defaults (k_1=1.2, b=0.75) are optimal.

Table 2
Retrieval model comparison, MRR@5 on the dev sets. BM25 uses Lucene defaults (k_1=1.2, b=0.75); only the most informative LM Dirichlet and LM Jelinek-Mercer rows are shown.

Model	EN	FR	DE	Avg
BM25 (k_1=1.2, b=0.75)	**0.4838**	**0.4887**	**0.3342**	**0.4356**
Classic (TF-IDF)	0.4130	0.3948	0.2715	0.3597
LM Dirichlet (μ=500)	0.4683	0.4652	0.3357	0.4231
LM Dirichlet (μ=1000)	0.4526	0.4434	0.3098	0.4019
LM Jelinek-Mercer (λ=0.5)	0.4784	0.4706	0.3220	0.4236

BM25 wins on the average by about 1.2 points over the strongest language-model variant and by 7.6 points over Classic TF-IDF. Classic is the worst row because it has no document-length normalisation, so long abstracts accumulate score regardless of how concentrated their relevant terms are. LM Dirichlet degrades as μ grows from 500 to 1000, since a stronger prior pulls scores towards the collection background and washes out the highly specific vocabulary that distinguishes scientific abstracts. LM Jelinek-Mercer is closer but never matches BM25 in any language.

We then swept $k_1 \in \{0.5, 0.8, 1.0, 1.2, 1.5, 2.0, 3.0\}$ against $b \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$. The b parameter dominates: moving from b=0 to b=0.75 adds about five points of average MRR@5 at every k_1, whereas at b=0.75 the entire k_1 column varies by less than 0.001. The optimum sits at k_1=1.0, b=0.75 with an average of 0.4357, within 0.0015 of the Lucene defaults. We lock in k_1=1.0, b=0.75 for the rest of the chapter.

5.3. Fielded Search and Query Construction

We now tune the query construction process by adjusting field boosts, deciding whether terms are mandatory or optional, and controlling the level of query pruning. Table 3 sweeps the title, abstract, and authors boosts in three parts (title with abstract fixed, abstract with title fixed, then authors on top of the winning ratio). Table 4 then prunes the query to the top-K tokens by IDF. The strictness of the Boolean query (OR vs. AND) is reported in prose, since the effect is too one-sided to need a table.

Parts A and B fix the title-to-abstract boost ratio. With the abstract boost held at 1.0, raising the title boost above 1.0 only hurts; with the title boost held at 2.0, raising the abstract boost up to 3.0 monotonically improves. The winning combination is a title boost of 2 and an abstract boost of 3, and what matters is the 2:3 ratio rather than the absolute values. Claims are tweet-style paraphrases of findings, not of titles, so the abstract is the primary evidence field, while the title still contributes a smaller but non-zero signal.

Part C adds the authors field on top of these settings, with the peak at an authors boost of 1.5 (0.4719 average), about 1.5 points above the no-authors row. Pushing higher hurts as mismatched proper nouns start to dominate. The DE column gains the most (about 2.4 points), consistent with German-translated

Table 3
Field boost sweeps on the EN/FR/DE dev sets, MRR@5. Part A varies the title boost with the abstract boost fixed at 1.0; Part B varies the abstract boost with the title boost fixed at 2.0; Part C adds the authors boost on top of the winning combination (title boost 2, abstract boost 3).

Boost weight	EN	FR	DE	Avg
Part A: title boost (abstract boost $= 1.0$*)*				
1.0	0.4997	0.5026	0.3577	0.4533
1.5	0.4929	0.4964	0.3475	0.4456
2.0	0.4838	0.4882	0.3352	0.4357
3.0	0.4635	0.4567	0.3159	0.4121
Part B: abstract boost (title boost $= 2.0$*)*				
1.0	0.4838	0.4882	0.3352	0.4357
1.5	0.4956	0.5000	0.3506	0.4487
2.0	0.4997	0.5026	0.3577	0.4533
3.0	0.4990	0.5066	0.3604	0.4553
Part C: authors boost (title boost $= 2.0$*, abstract boost* $= 3.0$*, topK*$=30$*)*				
0.5	0.4888	0.4938	0.3373	0.4400
1.0	0.4990	0.5066	0.3604	0.4693
1.5	**0.5172**	**0.5145**	**0.3841**	**0.4719**
2.0	0.5160	0.5130	0.3820	0.4715
3.0	0.5050	0.5020	0.3754	0.4608

tweets more often naming researchers than the English originals. We also swept a venue boost from 0.0 to 0.5 with no measurable effect.

We then varied the strictness of the Boolean query, marking the top-IDF terms as AND and the rest as OR. Even the lightest mixed variant (must$= 2$) drops the average to 0.087, and a full AND query collapses to 0.007. Claims paraphrase findings rather than quoting them, so any hard term constraint reduces recall heavily. We use pure OR throughout the rest of the chapter.

Table 4
IDF-based query pruning on top of the fielded query, MRR@5. Each claim is tokenised with the English analyzer; only the top-K tokens by IDF are passed as the query.

K	EN	FR	DE	Avg
all (no pruning)	0.4838	0.4882	0.3352	0.4357
10	0.4115	0.4189	0.2887	0.3730
15	0.4696	0.4658	0.3182	0.4179
20	0.4914	0.4798	0.3379	0.4364
30	**0.4956**	**0.4860**	**0.3395**	**0.4404**
40	0.4956	0.4860	0.3395	0.4404
50	0.4956	0.4860	0.3395	0.4404

For the IDF pruning sweep, the score climbs from K=10 to K=30 and then flattens: K=30, 40, and 50 are identical, since most claims contain fewer than 30 distinct content tokens after stop-word filtering. Smaller values are clearly harmful: at K=15 the average loses 2.3 points and at K=10 over six points, because genuine scientific terms get dropped along with ordinary content words. We set K=30. The locked-in lexical configuration is therefore BM25 (k_1=1.0, b=0.75) with field boosts (title 2, abstract 3, authors 1.5), OR mode, and the query pruned to the top 30 IDF tokens.

5.4. Failed Expansion Approaches

The principal challenge in this task is a vocabulary gap between informal, tweet-style claims and formal scientific abstracts. Section 3 describes four expansion strategies including a static synonym dictionary,

an LLM-based expander (Groq API), WordNet synonym injection, and pseudo-relevance feedback (PRF). The LLM-based and static dictionary strategies were not included in the systematic sweep reported here since the Groq API is rate-limited to approximately 30 requests per minute, making batch evaluation over thousands of claims impractical, and initial experiments with the static dictionary showed no measurable gain over the unexpanded baseline. For this reason, we evaluate WordNet and PRF, as they are self-contained and reproducible. We tested both on top of the analyzer-fixed configuration from Section 5.1, before the fielded boosting described in Section 5.3; this staging keeps the expansion experiments independent of the fielded tuning. In both cases, retrieval performance decreased as expansion grows.

Table 5
Query expansion experiments, MRR@5. All runs use the analyzer configuration fixed in Section 5.1.

Configuration	EN	FR	DE	Avg
Lexical baseline (no expansion)	0.4883	0.4901	0.3354	0.4379
WordNet, max 1 syn/word	0.4582	0.4352	0.3040	0.3991
WordNet, max 3 syn/word	0.4262	0.3993	0.2829	0.3695
WordNet, max 5 syn/word	0.4047	0.3759	0.2591	0.3466
PRF (3 docs, 3 terms)	0.4825	0.4763	0.3351	0.4313
PRF (5 docs, 3 terms)	0.4829	0.4790	0.3310	0.4310
PRF (10 docs, 10 terms)	0.4776	0.4692	0.3317	0.4262

WordNet expansion degrades the score monotonically: adding at most one synonym per word costs 3.9 points of average MRR@5, and at most five synonyms costs 9.1 points, leaving the system far below its unexpanded baseline. Three compounding factors explain this. First, WordNet is a general-vocabulary lexicon and covers very few of the precise scientific terms that appear in the corpus (“efficacy”, “mRNA”, “SARS-CoV-2”) so synonyms are generated mainly for the common words in the claim, not the discriminative ones. Second, the general synonyms produced by WordNet caused ambiguity. For example, WordNet may expand “system” in “immune system” with general-purpose synonyms that ignore the biomedical meaning of the full phrase, introducing tokens that match irrelevant documents. Third, every injected synonym increases query length, which reduces BM25’s IDF weighting on the original precise terms via length normalisation, making the genuine discriminative signal weaker rather than stronger.

PRF avoids the dictionary mismatch problem by extracting expansion terms from documents the system has already retrieved. The decrease in performance is smaller (0.7–1.2 points), but the direction is consistently negative. The root cause is topic drift. Since the corpus is dominated by COVID-19 and vaccine research, the top-retrieved documents for many health-related claims are often neighbouring papers on similar but distinct subtopics. Feeding their most frequent terms back into the query shifts the focus toward the broader topic cluster and away from the specific claim. BM25 length normalisation penalises the expanded query for the same reason as with WordNet, compounding the drift.

Both strategies are disabled in all later experiments. Although the vocabulary gap is real, our results suggest that expanding the lexical query with additional tokens is not an effective way to address it. Section 5.6 shows that dense fusion with BGE-M3 does close the gap, because embeddings place tweet-style paraphrases and formal abstracts close in a shared continuous space without any additional tokens entering the BM25 scoring path.

5.5. Ablation of the Lexical System

To isolate the contribution of each lexical component, we ablate them one at a time on top of the full best configuration locked in across Sections 5.1 to 5.3. Table 6 reports each ablation as a one-component change against that reference, plus a row that removes all three tunable components together, a row that adds the venue boost on top, and two rows that replace OR with stricter Boolean modes. All runs

share a single index, so differences come only from the component being changed.

Table 6
Ablation of the lexical system, MRR@5. The reference is the full configuration (BM25 k_1=1.0, b=0.75, field boosts title 2, abstract 3, authors 1.5, topK=30). Each row removes or changes exactly one component.

Configuration	EN	FR	DE	Avg	Δ Avg
Full best (reference)	0.5172	0.5145	0.3841	0.4719	n/a
− authorsBoost (set to 0)	0.5086	0.4973	0.3646	0.4568	−0.0151
− topKQueryTerms (use all terms)	0.5100	0.5131	0.3826	0.4686	−0.0034
− abstractBoost (set to 1)	0.4983	0.4885	0.3548	0.4472	−0.0247
− all three (simple baseline)	0.4874	0.4919	0.3323	0.4372	−0.0347
+ venueBoost = 0.5	0.5174	0.5155	0.3835	0.4721	+0.0002
queryMode = mixed (must= 2)	0.0752	0.1233	0.0635	0.0873	−0.3846
queryMode = must (all)	0.0151	0.0043	0.0026	0.0073	−0.4646

Ranked by impact, the abstract boost is the strongest single contributor: setting it back to 1 costs 2.5 points of average MRR@5. The authors boost is next at 1.5 points, and IDF pruning a smaller but still positive 0.3 points. Removing all three at once costs 3.5 points, slightly less than the 4.3-point sum of the individual deltas; this sub-additivity is expected, since the three components partially recover the same correct documents rather than improving orthogonal aspects of the ranking. The venue boost lands at $+0.0002$ on the average and is not adopted. The authors field is also visibly more useful on DE than on EN (-2.0 points vs. -0.9), consistent with German-translated tweets naming researchers more often than the English originals.

The two queryMode rows reproduce the collapse already seen in Section 5.3: switching just the two highest-IDF terms to AND drops the average to 0.087, and a fully AND query is essentially zero in every language. No claim text appears verbatim in any abstract, so the system depends on every term being optional. This confirms OR as the only viable Boolean mode and closes the lexical tuning.

5.6. Hybrid Retrieval: Lexical and Dense Fusion

On top of the lexical system fixed in the previous sections, we add a dense retrieval signal from a multilingual sentence encoder and combine the two scores. We compare two encoder families and two ways of combining the scores.

5.6.1. BGE-M3 Reranking and Fusion

BGE-M3 [12] is a multilingual sentence encoder that produces 1024-dimensional dense embeddings. For each query we encode the claim text and compare it against the pre-computed document embeddings by cosine similarity. We test two ways of incorporating this signal. In the reranking-only variant, the top-50 candidates from the lexical run are re-ordered by cosine similarity alone, with no BM25 score involved. In the fusion variant, the two scores are combined as

$$s = (1 - \alpha) \cdot \widetilde{s}_{\mathrm{BM25}} + \alpha \cdot \widetilde{s}_{\cos}, \tag{1}$$

where $\widetilde{s}_{\mathrm{BM25}}$ and $\widetilde{s}_{\cos}$ are the BM25 and cosine scores after min-max normalisation within each query's top-50 candidate set, and $\alpha \in [0, 1]$ controls the relative weight of the dense signal.

We swept α from 0.50 to 0.90 in steps of 0.02. The per-language peaks fall in the narrow range $[0.69, 0.73]$ (EN and FR peak near 0.73, DE near 0.69), and the MRR@5 curve is flat within 0.002 of its peak for every language between α=0.67 and α=0.75. We therefore adopt α=0.70 as a single practical setting that is within 0.002 of every per-language optimum.

Reranking alone adds only 1.2 points of average MRR@5. The lexical run already places the correct document near the top of the 50-candidate list for many queries, leaving little room for re-ordering

Table 7
Lexical baseline vs. BGE-M3 reranking and fusion, MRR@5. Reranking reorders the top 50 lexical hits by cosine; fusion combines $(1-\alpha) \cdot \text{BM25}_{\text{norm}} + \alpha \cdot \text{cosine}$ with both scores min-max normalised within each query's top-50.

Method	EN	FR	DE	Avg
Lexical baseline	0.5172	0.5145	0.3841	0.4719
BGE-M3 reranking only	0.5256	0.5280	0.3967	0.4834
BGE-M3 fusion (α=0.70)	**0.5768**	**0.5770**	**0.4331**	**0.5290**

to help. Fusion gains 5.7 points, a much larger margin, because it can promote candidates that BM25 ranked lower in the candidate list. A document whose precise vocabulary differs from the claim but whose meaning is close receives a low BM25 score and a high cosine score, and the combined score moves it into the top 5. The optimal α=0.70 assigns the embedding signal 2.3 times the weight of BM25, consistent with BM25 operating near its ceiling after the careful lexical tuning in the previous sections and the dense signal providing genuinely complementary evidence.

Role of translation in the hybrid system. Machine translation and multilingual dense retrieval play different roles in our pipeline. For the lexical BM25 component, translating French and German claims into English is useful because BM25 depends on exact term overlap and the document collection is mostly English. Without translation, many informative non-English query terms would not match the English index. By contrast, BGE-M3 does not require this normalization step, since it encodes English, French, and German claims in a shared multilingual embedding space. In the final hybrid system, we therefore use translation mainly to strengthen the sparse lexical signal, while relying on BGE-M3 to provide the main cross-lingual semantic signal. This also reduces the impact of possible translation errors: if a French or German claim contains biomedical terminology, named entities, compounds, or informal paraphrases that are translated imperfectly, the dense component can still retrieve semantically related abstracts from the original multilingual representation. Thus, translation is most helpful for BM25, whereas native multilingual retrieval is preferable for the dense stage.

We also experimented with a full-collection dense retrieval variant in which cosine similarity is computed over all 10,000 documents rather than restricted to the top-50 lexical candidates. Documents not in the lexical shortlist receive a BM25 score of zero, so for those candidates the fusion score reduces to the cosine term alone. At α=0.80 this configuration achieves a higher MRR@5 than the top-50 fusion because it can surface correct documents that the lexical run missed entirely. This full dense configuration is the one submitted as Run 2 and is used as the baseline for the cross-encoder experiments in Section 5.6.3.

5.6.2. SPECTER2 vs BGE-M3

BGE-M3 is a general-purpose multilingual encoder. We also tested SPECTER2 [13], a domain-specific encoder trained on scientific paper-to-paper citation proximity, on the hypothesis that its scientific specialisation would outweigh the loss of multilingual coverage. The hypothesis does not hold.

Table 8
SPECTER2 vs. BGE-M3 at their respective best fusion weights. Both use the same 50-candidate input lists from the lexical run.

Method	EN	FR	DE	Avg
Lexical baseline	0.5172	0.5145	0.3841	0.4719
SPECTER2 reranking only	0.4451	0.4233	0.3377	0.4020
SPECTER2 fusion (α=0.60)	0.5524	0.5475	0.4150	0.5050
BGE-M3 fusion (α=0.70)	**0.5768**	**0.5770**	**0.4331**	**0.5290**

SPECTER2 reranking alone drops 7 points below the lexical baseline (0.4020 vs. 0.4719), while BGE-M3 reranking adds 1.2 points. The gap can be explained by training objective and query distribution. Although SPECTER2 is domain-specific, its training objective focuses on scientific document similarity rather than informal-claim-to-abstract retrieval. This makes it less suitable for bridging tweet-style claims and formal scientific abstracts. BGE-M3 is trained on a broad mixture of text pairs that includes informal-to-formal cross-domain matching, making it a natural fit for bridging claim paraphrases to scientific abstracts. The optimal fusion weight for SPECTER2 is α=0.60, lower than the 0.70 required for BGE-M3, indicating that the SPECTER2 signal is weaker and needs BM25 weighted more heavily to avoid degrading the result.

As a diagnostic, we also tried pairing SPECTER2 document embeddings with BGE-M3 query embeddings by projecting them into a common 768-dimensional space using PCA. The cross-space cosine similarity yielded an MRR@5 of approximately 0.025, suggesting that the two embedding spaces are not directly compatible for cosine-based comparison. When fused with BM25, the best fusion weight dropped to 0.10, and the gain over the lexical baseline was between 0.0001 and 0.01 depending on the language, which is essentially negligible. This suggests that the mismatch is not limited to the query representation alone, since SPECTER2 document embeddings also appear to be less aligned with the requirements of claim-to-abstract retrieval. Overall, BGE-M3 with $\alpha = 0.70$ remains the preferred encoder.

5.6.3. Cross-Encoder Fine-Tuning

On top of the BGE-M3 fusion run we applied the fine-tuned cross-encoder described in Section 3. For this experiment the hybrid baseline uses the full dense retrieval configuration (cosine scored over all 10 000 documents, α=0.80), which achieves a higher MRR@5 than the top-50 fusion reported in Table 7. Table 9 compares this baseline against the task-specific fine-tuned model reranking its top-50 candidates.

Table 9
Cross-encoder fine-tuning on top of the full dense fusion run, MRR@5. The hybrid baseline uses BGE-M3 cosine over all 10 000 documents (α=0.80). Fine-tuned is `ms-marco-MiniLM-L-6-v2` after 2-epoch training on hard negatives from our retrieval pipeline, reranking the top-50 candidates.

Method	EN	FR	DE	Avg
Hybrid baseline (full dense fusion)	0.5875	0.5990	0.4615	0.5493
+ CE reranking (fine-tuned)	0.5606	0.6206	0.4783	0.5532

Fine-tuning improves French by 2.2 points and German by 1.7 points but degrades English by 2.7 points, for a net average gain of 0.4 points. The asymmetric effect is consistent with the relative size of the per-language training sets: the English training split is substantially larger and the base checkpoint was already well-calibrated for English-style claim-abstract pairs, so fine-tuning on comparatively fewer English examples provides little additional signal while introducing some overfitting. The smaller French and German training sets, by contrast, shift the model toward the specific vocabulary and phrasing of those languages, yielding clear gains. Given the mixed outcome, the fine-tuned reranker is not included in the primary submitted run.

Multilingual reranking imbalance. The language-dependent behavior of the fine-tuned cross-encoder suggests that a single reranker trained with uniform settings is not sufficient for a robust multilingual pipeline. Although fine-tuning improves French and German, it also shifts the model away from the stronger English calibration of the base checkpoint. Future versions should therefore treat reranking as a language-aware component rather than a single global post-processing step. Possible strategies include language-balanced sampling during fine-tuning, per-language validation and checkpoint selection, calibration of cross-encoder scores separately for English, French, and German, and the use of multilingual cross-encoder checkpoints. Another option is to train or tune separate

rerankers per language and combine them with the hybrid BM25+BGE-M3 score only when they improve the language-specific validation score.

5.7. Final Submitted Runs

Combining the winners of each phase gives the two configurations we submitted to the official evaluation. Run 1 is the pure lexical system, included as a reproducible baseline that depends only on the Lucene index and requires no GPU at query time. Run 2 is the primary submission: the full dense BGE-M3 fusion system (cosine scored over all 10 000 documents, α=0.80), which achieved the highest dev-set MRR@5 across all configurations evaluated.

Table 10
Submitted runs and dev-set MRR@5 per language.

Run	Description	EN	FR	DE	Avg
Run 1	Lexical only (best lexical config)	0.5172	0.5145	0.3841	0.4719
Run 2	Full dense hybrid: BGE-M3 fusion (α=0.80, all 10 000 docs)	0.5875	0.5990	0.4615	0.5493

Run 2 gains 7.7 points of average MRR@5 over Run 1 on the dev set. The improvement is consistent across all three languages: $+7.0$ points on EN, $+8.5$ on FR, and $+7.7$ on DE, so no single language is disproportionately favoured or penalised by the fusion. All hyperparameters (field boosts, k_1, b, α, and topK) were tuned only on the dev split.

Table 11 reports the official test-set scores for Run 2.

Table 11
Official test-set MRR@5 for the primary submitted run (Run 2: full dense BGE-M3 fusion, α=0.80, all 10 000 documents).

Run	EN	FR	DE	Avg
Run 2 (test)	0.5697	0.5738	0.4870	0.5435

The test-set average of 0.5435 is 0.6 points below the dev-set result of 0.5493, a small gap indicating the system generalised well to unseen data without significant overfitting. Per-language results are mixed: EN and FR drop from dev to test (-1.8 and -2.5 points respectively), while DE improves by 2.6 points. The DE gain is consistent with the dev set being a harder sample for German—its dev-set scores were lower throughout the tuning phase, so the test distribution appears somewhat more favourable for DE. Overall, the hybrid system transfers reliably across the unseen data, and the dense fusion gain over a pure lexical run is confirmed on held-out data.

Dev-set breakdown analysis

To see where the MRR@5 gains come from, Table 12 breaks the dev-set scores into hit rates and the fraction of queries for which no relevant document appeared in the top 100 (missing). Three configurations are compared: the lexical baseline (Run 1), the submitted full dense hybrid (Run 2: BGE-M3 cosine over all 10 000 documents, α=0.80), and the fine-tuned cross-encoder applied on top of Run 2.

Hybrid gains come from recall, not re-ranking. The lexical system misses any relevant document in the top 100 for about one in four queries on EN and FR, and one in three on DE. The hybrid cuts these missing rates to 14–19%, while also improving Hit@1 by 6–8 percentage points per language. Together these two effects explain the $+7$ to $+8.5$ pt MRR@5 gain: the dense signal finds documents BM25 misses entirely, and ranks them higher when BM25 does find them.

Table 12
Dev-set retrieval breakdown for the three evaluated configurations. The hybrid row corresponds to the submitted Run 2 (full dense BGE-M3 fusion, α=0.80, all 10 000 documents, avg MRR@5 = 0.5493). Hit@1 and Hit@5 are cumulative (percentage of queries with the correct document ranked ≤ 1 and ≤ 5, respectively); Missing is the fraction of queries where the correct document does not appear in the top 100.

System	Lang	MRR@5	Hit@1 (%)	Hit@5 (%)	Missing (%)
Lexical (BM25)	EN	0.5172	46.6	59.5	26.1
	FR	0.5145	44.9	61.0	24.5
	DE	0.3841	32.9	48.2	35.2
Full dense hybrid (α=0.80)	EN	0.5875	53.0	67.7	14.6
	FR	0.5990	53.7	69.5	14.0
	DE	0.4615	40.2	55.4	19.4
Cross-encoder (not submitted)	EN	0.5606	49.5	66.9	23.3
	FR	0.6206	56.7	70.8	22.8
	DE	0.4783	42.2	57.0	37.0

Table 13
Final versus exploratory components in the system.

Component	Role in submitted system	Decision
BM25 lexical retrieval	Submitted as Run 1 and used in Run 2	Kept
BGE-M3 dense fusion	Submitted as Run 2, the primary system	Kept
Query expansion	Exploratory only	Discarded
SPECTER2 retrieval	Exploratory only	Discarded
Fine-tuned cross-encoder	Exploratory only	Not used in primary run

Cross-encoder: precision up, recall down, asymmetric across languages. The CE only reranks the top-50 hybrid candidates, so any document at hybrid ranks 51–100 is effectively discarded. This raises the missing rate by $+8.6$ pp for EN, $+8.8$ pp for FR, and $+17.6$ pp for DE. Despite this recall hit, the CE still comes out ahead on FR ($+2.2$ pts MRR@5) and DE ($+1.7$ pts), because it promotes more queries to rank 1 in those languages ($+3.0$ pp and $+2.1$ pp Hit@1). For EN it is the opposite: the CE pushes 138 queries down from rank 1, costing -2.7 pt MRR@5, which is the primary reason we did not submit the fine-tuned reranker.

German is the hardest language. DE trails EN and FR by 13–14 MRR@5 points in every configuration. Its lexical missing rate (35.2%) is about ten percentage points worse than EN and FR, and the CE pushes it back almost to that level (37.0%), suggesting that the hybrid's recall gain on DE is fragile once the candidate list is cut to 50. The hybrid is the configuration that helps DE the most in absolute terms (-15.8 pp missing), which fits with multilingual BGE-M3 doing a better job at cross-lingual matching than the lexical index can achieve alone.

Final versus exploratory components. Table 13 clarifies which components were part of the submitted system and which were only explored during development.

5.8. Statistical Analysis

To verify that the MRR@5 differences reported above are not due to sampling variance, we performed a two-way ANOVA on the per-query RR@5 values, blocking on topics [15]. The two-way design accounts for query-level difficulty, which increases statistical power compared to a one-way test. The three configurations (Lexical, Hybrid, Cross-Encoder) act as treatment factors; each query constitutes a block that is evaluated by all three systems simultaneously. Figure 1 shows the distribution of the first relevant

rank for each language and Figure 2 reports the two-way ANOVA F-statistics and the post-hoc Tukey HSD pairwise comparisons ($\alpha = 0.05$).

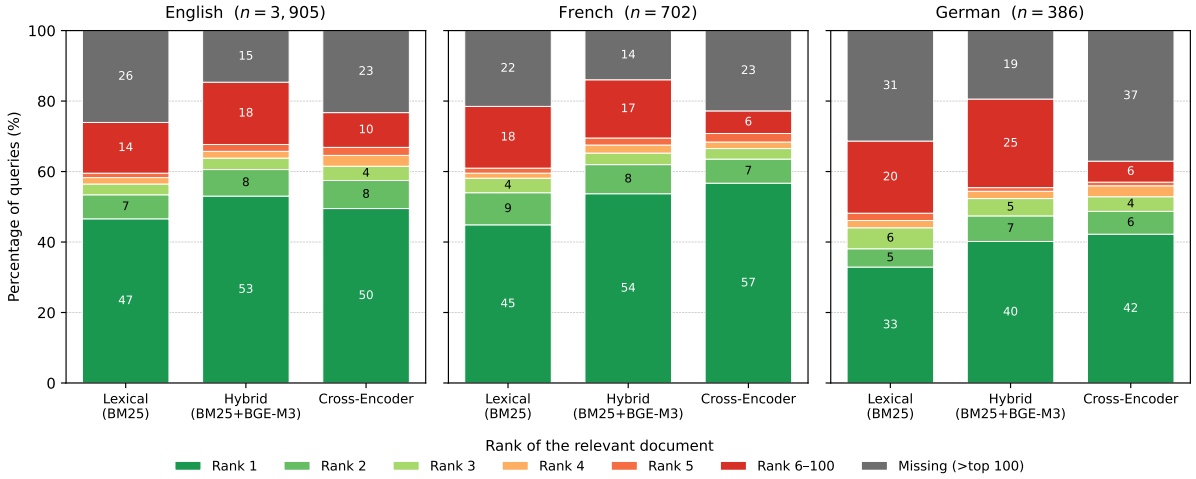

Figure 1: Rank of the relevant document per query, dev set. Bars split queries into seven buckets: ranks 1–5, ranks 6–100 (found but outside the top 5, a re-ranking failure), and Missing (not in the top 100, a recall failure). The cross-encoder is applied only to a short candidate list from the hybrid run, so any document outside that list is counted as Missing here.

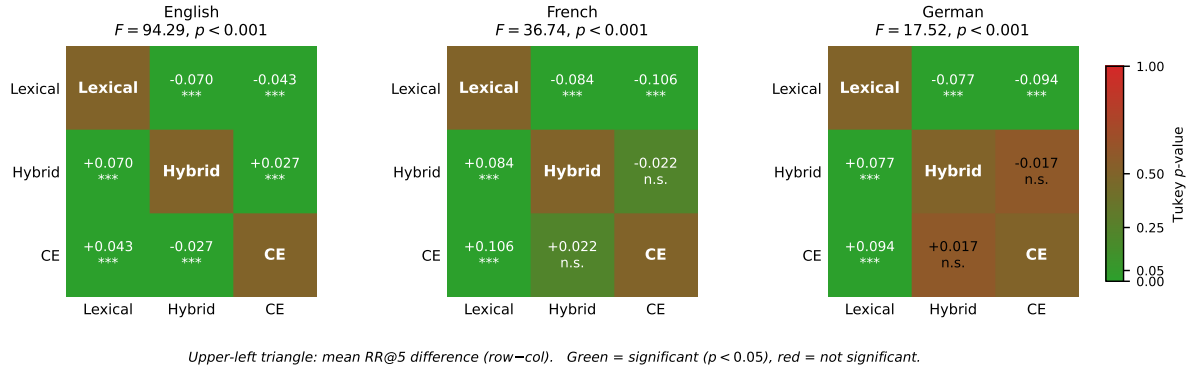

Figure 2: Two-way ANOVA results and Tukey HSD post-hoc pairwise comparisons per language. Each off-diagonal cell shows the mean RR@5 difference (row − col) and the significance marker. Cell colour encodes the Tukey p-value: green indicates a significant difference ($p < 0.05$), red a non-significant one.

Table 14
Two-way ANOVA results per language. All three languages reject H_0 at $\alpha = 0.05$.

Language	n queries	F	p-value	α	Decision
English	3905	94.29	< 0.001	0.05	Reject H_0
French	702	36.74	< 0.001	0.05	Reject H_0
German	386	17.52	< 0.001	0.05	Reject H_0

Table 14 reports the F-statistics and p-values. All three languages reject the null hypothesis (H_0: all systems perform equally) well below the 0.001 threshold, confirming that at least one pair of configurations differs significantly on every language partition. The Tukey HSD post-hoc test localises the differences:

- **Lexical vs. Hybrid** is significant in all three languages ($p < 0.001$), confirming that the BGE-M3 fusion gain is not attributable to chance.

- **Lexical vs. Cross-Encoder** is also significant across all languages ($p < 0.001$), placing the fine-tuned reranker consistently above the pure lexical baseline.
- **Hybrid vs. Cross-Encoder** is significant only in English ($p < 0.001$, Hybrid > Cross-Encoder by Δ=0.027). In French and German the cross-encoder leads by approximately 0.02 MRR@5 but the difference is not statistically significant ($p = 0.224$ and $p = 0.582$ respectively), so neither system is conclusively better on those languages.

In summary, the dense fusion component provides a robust, statistically confirmed improvement over the lexical baseline in every language. The cross-encoder reranking offers a statistically indistinguishable trade-off relative to hybrid retrieval across most of the evaluation, with its advantage in French and German and its disadvantage in English cancelling out at the aggregate level.

5.9. Test-Set Statistical Analysis

We repeated the two-way ANOVA and Tukey HSD analysis on the test set with the same three configurations as on dev: the lexical baseline (Run 1), the BGE-M3 hybrid (α=0.80, Run 2), and the fine-tuned cross-encoder of Section 5.6.3 applied on top of Run 2. Figure 3 shows the first relevant rank distributions by language and Figure 4 the Tukey HSD heat-maps.

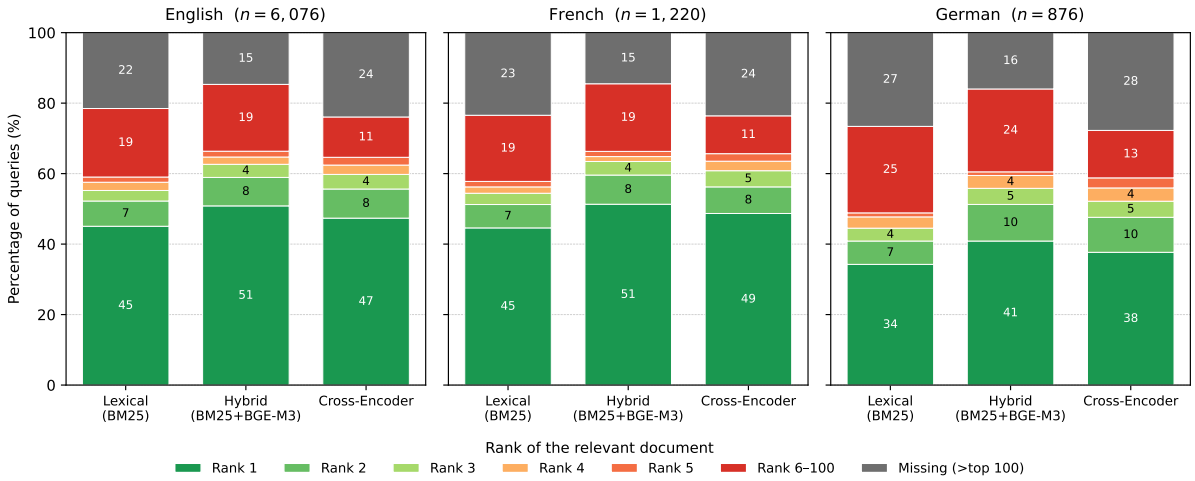

Figure 3: Rank of the relevant document per query, test set. Bucket conventions follow Figure 1.

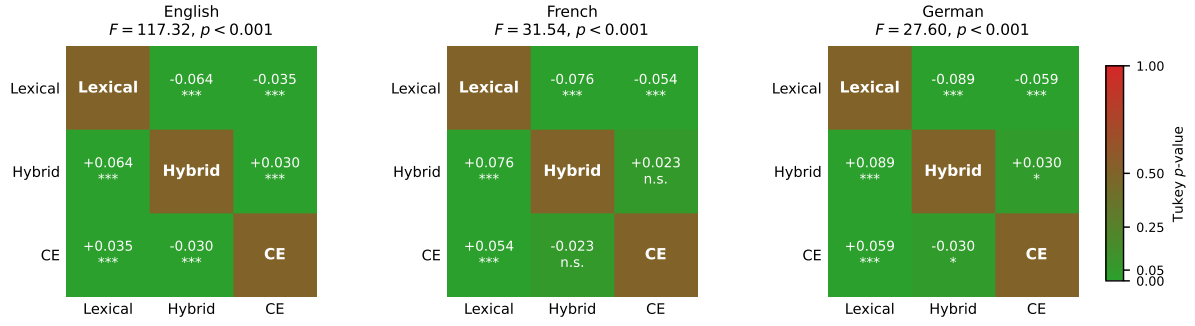

Figure 4: Two-way ANOVA and Tukey HSD post-hoc comparisons on the test set (fine-tuned CE). Cell colour and annotation conventions are the same as Figure 2.

Table 15 confirms that at least one system differs significantly on every language partition. The Tukey HSD post-hoc results are:

Table 15
Two-way ANOVA results on the test set. All three languages reject H_0 at $\alpha = 0.05$.

Language	n queries	F	p-value	α	Decision
English	6076	117.32	< 0.001	0.05	Reject H_0
French	1220	31.54	< 0.001	0.05	Reject H_0
German	876	27.60	< 0.001	0.05	Reject H_0

- **Lexical vs. Hybrid** is significant in all three languages ($p < 0.001$), replicating the dev-set finding on held-out data with substantially larger query sets.
- **Lexical vs. Cross-Encoder** is significant in all three languages ($p < 0.001$ for EN and FR; $p = 0.000005$ for DE), confirming that the fine-tuned CE consistently outperforms the pure lexical system.
- **Hybrid vs. Cross-Encoder** is significant in English ($p < 0.001$, Hybrid $>$ CE by Δ=0.030) and German ($p = 0.035$, Hybrid $>$ CE by Δ=0.030). In French the difference is not significant ($p = 0.054$, Hybrid $>$ CE by Δ=0.023), indicating that fine-tuned CE and hybrid are statistically equivalent on that language.

Compared with the dev-set analysis, the test set confirms the dominance of hybrid over lexical and extends the Hybrid $>$ CE finding from English alone to English and German. The fine-tuned model therefore does not improve over the hybrid in any language on the test partition, which is consistent with the per-query MRR@5 values (EN: 0.5398 vs. 0.5697; FR: 0.5509 vs. 0.5738; DE: 0.4567 vs. 0.4870). The result reinforces the conclusion that the BGE-M3 fusion is the strongest and most reliable configuration across all languages and both evaluation partitions.
Statistical metrics defined. SS (Sum of Squares) measures variability attributable to each source; df (Degrees of Freedom) is the number of independent values free to vary; MS (Mean Square) $= SS/df$ standardises the variance estimate. The F-statistic is the ratio of the Systems MS to the Error MS, and determines whether observed performance differences are statistically significant or due to chance.

6. Conclusions and Future Works

In this project, we developed a configurable multi-stage Information Retrieval system for Task 1 of the CLEF 2026 CheckThat! Lab, focused on retrieving scientific sources for social media claims in English, French, and German. The system was designed as a modular pipeline separating parsing, text cleaning, text analysis, indexing, retrieval, and reranking, which allowed us to test different configurations systematically.

The results show that a carefully tuned lexical system remains a strong baseline for this task. Translating all inputs into English simplified the multilingual setting and enabled the use of a single English analyzer pipeline based on the Standard tokenizer, KStem, and an English stop list. BM25 was the strongest lexical ranking model among the alternatives tested, while fielded retrieval further improved performance by giving different weights to the title, abstract, and authors fields. In particular, the abstract field proved especially important, since social media claims usually paraphrase scientific findings rather than paper titles.

Our experiments also show that strict Boolean matching and standard lexical query expansion are not well suited to this task. AND constraints reduced recall sharply because claims and abstracts rarely share exact wording. Similarly, WordNet expansion and Pseudo-Relevance Feedback degraded performance due to ambiguity, vocabulary mismatch, and topic drift.

The best results were obtained by combining lexical and semantic evidence. Fusing BM25 scores with BGE-M3 dense similarity improved the average MRR@5 from 0.4719 for the best lexical system to 0.5493 for the hybrid system. This suggests that dense embeddings are most useful as a complementary signal to BM25, helping bridge the vocabulary and style gap between informal social media claims and formal scientific abstracts.

Table 16
Detailed variance decomposition (SS, df, MS) for the Two-way ANOVA on Development and Test sets.

Dataset	Language	Source of Variation	SS	df	MS
Dev set	English	Systems	9.8397	2	4.9198
		Topics (Queries)	2084.0922	3904	0.5338
		Error	407.4079	7808	0.0522
	French	Systems	4.4135	2	2.2067
		Topics (Queries)	352.4729	701	0.5028
		Error	84.2036	1402	0.0601
	German	Systems	1.9489	2	0.9744
		Topics (Queries)	201.7964	385	0.5241
		Error	42.8167	770	0.0556
Test set	English	Systems	12.6469	2	6.3234
		Topics (Queries)	3226.8517	6075	0.5312
		Error	654.9016	12150	0.0539
	French	Systems	3.7525	2	1.8762
		Topics (Queries)	636.5225	1219	0.5222
		Error	145.0290	2438	0.0595
	German	Systems	3.5873	2	1.7936
		Topics (Queries)	424.0025	875	0.4846
		Error	113.7323	1750	0.0650

Our experiments with task-specific cross-encoder fine-tuning showed mixed results: the fine-tuned model improved French and German MRR@5 but degraded English, suggesting that reranking is language-dependent and sensitive to the size and balance of the training data. This indicates that a single globally fine-tuned cross-encoder is not sufficient for a robust multilingual retrieval pipeline. Future work will therefore investigate language-aware reranking strategies, including language-balanced sampling, per-language validation and checkpoint selection, separate score calibration for English, French, and German, and multilingual cross-encoder checkpoints. We also plan to test whether separate rerankers for each language can improve French and German without degrading the already strong English baseline. In addition, future efforts will explore selective Pseudo-Relevance Feedback to mitigate topic drift, adaptive hybrid ranking fusion, improved translation and error analysis for lower-performing languages such as German, and systematic evaluation of the efficiency-effectiveness trade-offs of computationally expensive reranking models.

7. Declaration on Generative AI

During the preparation of this work, Grammarly was used for grammar and spelling checking. Chat- GPT and Gemini were used for rephrasing and sentence polishing to improve clarity and readability. ChatGPT was also used to convert parts of the manuscript into LATEX code and as a formatting assistance. In addition, Google Translate was used for text translation and subsequent language polishing. All content was reviewed and edited by the authors, who take full responsibility for the final manuscript.

References

[1] 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.

[2] The clef 2026 checkthat! lab, https://clef2026.clef-initiative.eu/labs/checkthat/, 2026. Accessed: 2026-05-24.

[3] T. Elsayed, P. Nakov, A. Barrón-Cedeño, M. Hasanain, R. Suwaileh, G. Da San Martino, P. Atanasova, Overview of the CLEF-2019 CheckThat!: Automatic Identification and Verification of Claims, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019), 2019, pp. 301–321.

[4] 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: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Eleventh International Conference of the CLEF Association (CLEF 2020), 2020, pp. 215–236.

[5] P. Nakov, G. Da San Martino, T. Elsayed, A. Barrón-Cedeño, R. Míguez, S. Shaar, F. Alam, F. Haouari, M. Hasanain, W. Mansour, B. Hamdan, Z. S. Ali, N. Babulkov, A. Nikolov, G. K. Shahi, J. M. Struß, T. Mandl, M. Kutlu, Y. S. Kartal, Overview of the CLEF–2021 CheckThat! Lab on Detecting Check-Worthy Claims, Previously Fact-Checked Claims, and Fake News, in: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Twelfth International Conference of the CLEF Association (CLEF 2021), 2021, pp. 264–291.

[6] S. Hafid, Y. S. Kartal, S. Schellhammer, K. Boland, D. Dimitrov, S. Bringay, K. Todorov, S. Dietze, Overview of the CLEF-2025 CheckThat! lab task 4 on scientific web discourse, in: Working Notes of CLEF 2025 - Conference and Labs of the Evaluation Forum, CEUR-WS.org, 2025, pp. 722–732.

[7] C. D. Manning, P. Raghavan, H. Schütze, Introduction to Information Retrieval, Cambridge University Press, Cambridge, UK, 2008.

[8] P. J. Sager, A. Kamaraj, B. F. Grewe, T. Stadelmann, Deep retrieval at checkthat! 2025: Identifying scientific papers from implicit social media mentions via hybrid retrieval and re-ranking, arXiv (2025). doi:10.48550/arxiv.2505.23250.

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

[10] M. McCandless, E. Hatcher, O. Gospodnetić, Lucene in Action, 2nd ed., Manning Publications Co., NY, USA, 2010.

[11] J. J. Rocchio, Relevance Feedback in Information Retrieval, in: G. Salton (Ed.), The SMART Retrieval System. Experiments in Automatic Document Processing, Prentice–Hall, Inc., Englewood Cliff, New Jersey, USA, 1971, pp. 313–323.

[12] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation, arXiv preprint arXiv:2309.07597 (2024).

[13] A. Singh, M. D'Arcy, A. Cohan, D. Downey, S. Feldman, SciRepEval: A Multi-Format Benchmark for Scientific Document Representations, in: Proceedings of EMNLP 2023, 2023.

[14] S. Schellhammer, Checkthat! 2026 task 1: Source retrieval for scientific web claims dataset, https://huggingface.co/datasets/sschellhammer/CT26_Task1_SourceRetrievalForScientificWebClaims, 2026. Accessed: 2026-XX-XX.

[15] T. Sakai, Topic set size design, Information Retrieval Journal 19 (2016) 256–283.