Boyes Boys at CheckThat! 2026: Go big or go home

CheckThat! Task 1 – Source Retrieval for Scientific Web Claims

Mattias Kvist[1,†], Eduardo Nazário[1,†], Derin Suzener[1,†] and Osvald Svenaeus[1,*,†]

[1]KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

This paper describes the system developed by team Boyes Boyes, to the CLEF 2026 CheckThat! Task 1 on source retrieval for scientific web claims. The task asks systems to retrieve the scientific paper implicitly or explicitly referenced by a short social media post. We build a multi-stage retrieval pipeline that combines an optimized sparse retriever, a large instruction-prompted dense retriever based on `microsoft/harrier-oss-v1-27b`, supervised Random Forest fusion, and cross-encoder reranking. On the development set, the best configuration combines Harrier dense retrieval, optimized sparse retrieval, Random Forest fusion, and `Qwen/Qwen3-Reranker-8B`, reaching 0.7459 average MRR@5 across English, German, and French. On the official test leaderboard, the system placed third overall with an average score of 0.7464, third for English with 0.7571, third for German with 0.7070, and second for French with 0.7752.

Keywords

scientific claim retrieval, hybrid retrieval, sparse retrieval, dense retrieval, random forest, reranking, CheckThat

1. Introduction

The CLEF 2026 CheckThat! Task 1, Source Retrieval for Scientific Web Claims, focuses on retrieving the scientific paper referred to by a short web claim or social media post [1, 2]. The input posts are often informal, multilingual, noisy, and much shorter than the scientific metadata they must be matched against. The target collection, by contrast, consists of academic paper metadata such as titles, authors, abstracts, and venues. This creates a substantial lexical and stylistic gap between the query and the relevant document.

Our submission treats the task as a multi-stage information retrieval problem. First-stage retrievers generate candidate papers with complementary strengths: a sparse BM25-based branch preserves exact lexical signals, while a large dense embedding model captures semantic and cross-lingual similarity. The candidate lists are then merged using either reciprocal rank fusion (RRF) or a learned Random Forest fuser, and the top candidates are reranked with cross-encoder reranking models.

The final system achieved strong performance on both development and official test data. On the official leaderboard it ranked third overall, with particularly strong French performance where it ranked second. The main contribution of the system is an empirical comparison of retrieval scale, learned fusion, and reranking choices for scientific claim source retrieval.

2. Task and Data

Task 1 evaluates whether a system can return the correct scientific paper in the top five retrieved results for each query. The primary metric is MRR@5, computed from the rank position of the first relevant document. Since each query has a single target paper, MRR@5 directly measures whether the system places that target paper near the top of the ranked list.

Table 1 summarizes the available query data. The paper corpus contains 10,000 candidate documents represented through metadata fields, including title, authors, abstract, and venue. The multilingual

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

*Corresponding author.

†All authors contributed equally.

makv@kth.se (M. Kvist); edfn@kth.se (E. Nazário); suzener@kth.se (D. Suzener); osvalds@kth.se (O. Svenaeus)

Table 1
Training and development data used during system development.

Language	Train queries	Development queries
English	14,977	3,905
German	1,460	386
French	2,807	702

query set is especially challenging for sparse retrieval because German and French posts must often be matched against English scientific metadata.

For a query set Q, we compute:

$$\text{MRR@5} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \cdot \mathbb{I}(rank_i \leq 5), \tag{1}$$

where $rank_i$ is the rank of the correct paper for query i. We report per-language MRR@5 and the unweighted average across English, German, and French.

3. Related work

Source retrieval for scientific claims is an emerging sub-task that combines elements of argument mining and traditional academic literature search. The official CheckThat! getting-started notebook provides a dense bi-encoder baseline using `intfloat/multilingual-e5-large` from the E5 embedding model family [3]. It encodes tweets and papers independently and ranks papers by embedding similarity, reaching 0.4446 average MRR@5 on the development set.

A highly relevant benchmark for our approach is the system proposed by the "Deep Retrieval" team during the CLEF CheckThat! 2025 competition (Subtask 4b). Sager et al. [4] successfully utilized a hybrid architecture consisting of standard BM25 lexical retrieval and a FAISS vector store driven by a fine-tuned `INF-Retriever-v1` model. By fusing 30 sparse and 100 dense candidates and feeding them into the `bge-reranker-v2-gemma` LLM cross-encoder, they achieved an MRR@5 of 76.46% on the 2025 development set.

4. System Overview

Figure 1 shows the overall architecture. The system uses separate sparse and dense candidate generation branches. Their outputs are merged into a single candidate set, scored by a fusion method, and then reranked by a cross-encoder reranker. The final output is the top five ranked paper identifiers.

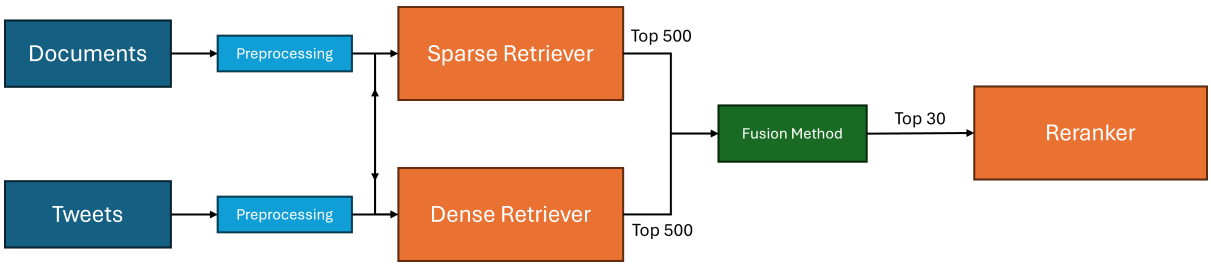

Figure 1: Overview of the hybrid retrieval and reranking pipeline.

The system was developed through a 15-configuration ablation study. We first optimized a sparse retriever, then compared smaller fine-tuned dense retrieval with a larger zero-shot dense retriever, and

finally evaluated fusion and reranking variants. This design kept the pipeline modular enough to test whether improvements came from candidate generation, fusion, or final reranking.

5. Retrieval and Fusion

5.1. Sparse Retrieval

The sparse branch is based on BM25 [5]. A direct BM25 baseline is effective when a tweet shares rare terms, names, or numerical details with the target paper, but it struggles with informal wording and cross-lingual mismatch. We therefore optimized preprocessing and indexing specifically for this task.

The most important intervention was translating all tweets to English using the Python package `deep-translator`'s `GoogleTranslator` (auto-detected source, English target). Tweets were preprocessed by removing URLs, emojis, and hashtags before translation; after translation, original terms longer than five characters that were not stopwords were appended to preserve names, numbers, and technical terms in their lexical form.

Text was lowercased, stripped of punctuation, filtered with combined EN/DE/FR NLTK stopwords, stemmed with LancasterStemmer, and augmented with adjacent-stem bigrams. Documents were indexed by repeating the title three times and venue twice before the abstract, up-weighting these fields in BM25+. Tuned parameters were $k_1 = 2.5$, $b = 0.85$ (vanilla: 1.5, 0.75).

For English queries, a term co-occurrence graph (window 5) propagated weight over two diffusion steps (decay 0.65, 6 neighbors). Additionally, PRF re-scored the top 7 BM25 documents, collected the 8 most frequent terms, and added their relative frequency weighted by 0.85 to matching documents.

Candidate configurations for the sparse retrieval pipeline were initially explored using an autonomous, LLM-driven experimentation loop powered by an AutoResearch agent framework [6] and structured using the AutoResearch system skill specification [7]. A detailed description of the process is provided in appendix B.

To ensure the robustness of the discovered configuration and prevent overfitting to the development slice. We extracted the promising components identified by the agent, subjected them to manual validation and retested some discarded avenues. Controlled experiments were conducted using cross-folding, keeping only those modifications that showed statistical significance. Because the sparse retriever acts strictly as a candidate generator feeding a downstream neural reranker, maintaining this high-recall optimization boundary during the initial stage was vital to overall system performance.

5.2. Dense Retrieval

We evaluated two dense retrieval directions. The first used `BAAI/bge-m3` [8], fine-tuned with contrastive learning and Low-Rank Adaptation (LoRA) [9]. For each labeled query, the tweet was used as the anchor, the target paper metadata as the positive, and BM25-retrieved non-matching papers as hard negatives. This improved over the untuned dense retriever, but the strongest dense results came from scaling the model.

The final dense branch used `microsoft/harrier-oss-v1-27b` [10]. Rather than fine-tuning Harrier, we used it zero-shot with a task-specific instruction prompt:

```
Instruct: Retrieve the implicitly referenced scientific
article\nQuery: <TWEET_TEXT>
```

Documents were represented with title, abstract, venue, and author metadata and embedded once. Queries were embedded separately and ranked by cosine similarity against the cached document embedding matrix.

5.3. Fusion

The sparse and dense branches provide complementary evidence. Sparse retrieval preserves exact lexical matches, while dense retrieval captures semantic similarity. We compared two ways to combine

their ranked lists into one.

The first was reciprocal rank fusion, an unsupervised rank-combination method:

$$\text{RRF}(d) = \sum_{r \in R} \frac{1}{k + rank_r(d)}, \tag{2}$$

where $rank_r(d)$ is the rank of document d in retriever r and k is a smoothing constant.

The second was supervised Random Forest fusion [11]. The fuser was trained on the training set to estimate the relevance probability of query-document candidate pairs from sparse and dense scores and ranks. At inference time, candidate papers for a query were sorted by the predicted probability of relevance. This avoided treating retrieval as a 10,000-class classification problem and instead learned how to weight retrieval signals for candidate pairs. Parameters and description can be found in appendix A.

6. Reranking

The final stage reranks the top fused candidates with cross-encoder rerankers. We evaluated `nvidia/llama-nemotron-rerank-1b-v2` [12], `jinaai/jina-reranker-v3` [13], and `Qwen/Qwen3-Reranker-8B` [14]. Earlier experiments also used `BAAI/bge-reranker-v2-gemma`.

Due to compute limits, only the top 30 fused candidates per query were reranked. Reranker inputs were capped at 2,048 tokens. This value was chosen after inspecting query-document input lengths: the distribution had a right skew, and 2,048 tokens covered most typical title-and-abstract cases without the memory cost of a 4,096-token setting.

The Qwen3 reranker used an instruction-aware scoring format. Each query-document pair was framed as a relevance decision with the instruction:

> `Given a scientific claim or social media post, retrieve the scientific paper that the claim refers to or is supported by.`

We scored the pair by comparing the final-token logits for the tokens `yes` and `no`, applying a two-way softmax, and using the `yes` probability as the relevance score.

7. Experimental Setup

All development experiments were evaluated on the official development splits for English, German, and French. We used the official CheckThat! baseline based on `intfloat/multilingual-e5-large` [3] as a reference point. Larger model inference used cloud GPU infrastructure with high-memory A100, H100, H200 and B200 instances, primarily because Harrier 27B and the larger reranking models were not feasible to run on lightweight local hardware.

The ablation study was designed to isolate the effect of each pipeline stage. It compared vanilla and optimized sparse retrieval, untuned and fine-tuned dense retrieval, Harrier dense retrieval, RRF and Random Forest fusion, and three final rerankers. Unless otherwise stated, reported development numbers are MRR@5.

8. Final Run

For the final run of the test set some small tweaks were made to ensure maximum score using the best performing pipeline, these included:

1. Extended top-K candidates from random forest fuser to final reranker from 30 to 200 for German, and to 100 for English and French.
2. Extended the reranker context window size from 2048 to 8192 tokens.

Although the gains were modest, increasing the number of reranked candidates consistently improved development set performance, motivating the configuration used for the final submission (Table 3).

The extension of the context window, however, actually proved to yield a marginally lower score for FR and DE, as shown on Table 4.

9. Results

Table 2
Development-set ablation results measured by MRR@5.

Configuration	EN	DE	FR	Avg.
CheckThat! baseline	0.4987	0.3767	0.4584	0.4446
Vanilla BM25 sparse retriever	0.4991	0.1973	0.2634	0.3199
Optimized sparse retriever	0.5514	0.3987	0.5511	0.5004
BGE-M3 dense retriever	0.5244	0.4129	0.5328	0.4900
Fine-tuned BGE-M3 dense retriever	0.5728	0.5044	0.5892	0.5555
Harrier 27B dense retriever	0.6943	0.5892	0.7051	0.6628
Hybrid + RRF fusion	0.6325	0.5103	0.6440	0.5956
Hybrid + Random Forest fusion	0.7000	0.5937	0.7127	0.6688
Hybrid + RF + Nemotron reranker	0.7391	0.6244	0.7343	0.6993
Hybrid + RRF + Nemotron reranker	0.7311	0.6166	0.7273	0.6917
Harrier + Nemotron reranker	0.7388	0.6190	0.7347	0.6975
Harrier + Qwen3 8B reranker	0.7570	0.6877	0.7801	0.7416
Hybrid + RF + Qwen3 8B reranker	**0.7584**	**0.6943**	**0.7850**	**0.7459**
Hybrid + RRF + Qwen3 8B reranker	0.7474	0.6896	0.7794	0.7388
Hybrid + RF + Jina reranker	0.6952	0.6203	0.7026	0.6727
Hybrid + RRF + Jina reranker	0.6881	0.6089	0.6989	0.6653

Table 2 shows that the best development configuration combines optimized sparse retrieval, Harrier dense retrieval, Random Forest fusion, and Qwen3 8B reranking. It improves average MRR@5 from 0.4446 for the CheckThat! baseline to 0.7459. The largest gains come from replacing smaller dense retrieval with Harrier and adding Qwen3 reranking on top of the fused candidate set.

Table 3
Impact of the number of reranking candidates on MRR@5.
Hybrid + RF + Qwen3 8B reranker run on the development-set.

# Docs	EN	FR	DE
50	0.7613	0.7872	0.7009
100	**0.7621**	0.7882	0.7044
200	–	**0.7913**	**0.7061**

Table 3 shows that increasing number of reranking candidates increases the MRR@5 consistently for the development-set using the final pipeline. English was left out due to budget constraints.

Table 4
Impact of Maximum Reranker Token Length on MRR@5.
Hybrid + RF + Qwen3 8B reranker run on the development-set, 30 reranking candidates.

lang	2048	8192
EN	0.7491	**0.7493**
FR	**0.7829**	0.7790
DE	**0.7038**	0.6974

Table 4 shows that increasing Token length for the reranker has no improvement on MRR@5.

Table 5
Official test leaderboard results for the submitted system.

Evaluation	Rank	Score
Overall average	3rd	0.7464
English	3rd	0.7571
German	3rd	0.7070
French	2nd	0.7752

Table 5 reports the official test leaderboard performance. The system placed third overall with an average score of 0.7464. It ranked third on English and German, and second on French. The test scores are close to the best development average, suggesting that the final configuration generalized well from the development split to the hidden test evaluation.

10. Discussion

The ablation study highlights several practical lessons for source retrieval from scientific web claims.

Sparse retrieval remains useful. Optimized sparse retrieval improved average development MRR@5 from 0.3199 for vanilla BM25 to 0.5004. The main driver was translation of German and French queries into English. Translation did not make BM25 semantic, but it removed the most severe cross-lingual lexical mismatch and allowed field weighting, normalization, and BM25 tuning to become effective. The Harrier dense-only plus Nemotron reranker ablation reached 0.6975, close to the hybrid Random Forest plus Nemotron result of 0.6993. With Qwen3 reranking, the dense-only configuration reached 0.7416—the second best overall, only 0.0043 below the full Hybrid+RF+Qwen3 setup at 0.7459. Harrier dense retrieval alone is already highly competitive when paired with the strongest reranker; the sparse branch adds only a small additional gain, primarily through Random Forest fusion of strong lexical signals.

Model scale outperformed smaller-model fine-tuning. Fine-tuning BGE-M3 with hard negatives improved average development MRR@5 to 0.5555. However, zero-shot Harrier 27B reached 0.6628 without task-specific fine-tuning. This suggests that, for this dataset, the representational scale and instruction-following behavior of the larger embedding model were more valuable than adapting a smaller model. The tradeoff is substantial compute cost. To fully support this claim, further experiments are needed.

Learned fusion was more reliable than static fusion. Random Forest fusion outperformed RRF before reranking and remained better after reranking for the strongest Qwen3 and Nemotron configurations. Figure 2 shows that dense-rank features dominate the fuser, but sparse features retain non-zero importance. This supports the interpretation that sparse evidence helps in cases involving exact names, numbers, or rare terms.

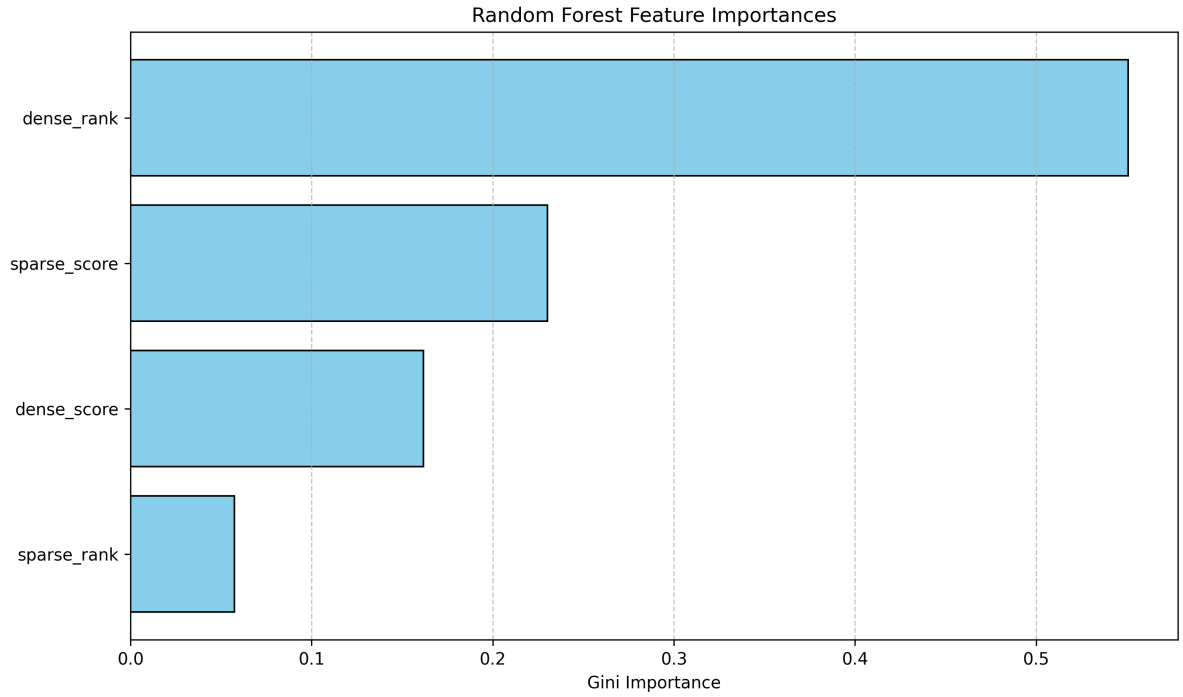

Figure 2: Random Forest feature importance measured by Gini impurity reduction.

Reranking produced the strongest final improvement. Hybrid Random Forest fusion without reranking reached 0.6688 average development MRR@5, while Qwen3 8B reranking raised the score to 0.7459. Qwen3 outperformed Nemotron and Jina in the final configurations. The reranking stage was still limited by cost for the ablation study: only 30 candidates were reranked per query, and inputs were capped at 2,048 tokens. Subsequent investigation show that a higher number of reranking candidates yields a slight performance improvement whilst the importance of increased context remains unclear.

11. Conclusion and Future Work

The final pipeline combines optimized BM25 retrieval, Harrier 27B dense retrieval, supervised Random Forest fusion, and Qwen3 8B reranking. It reached 0.7459 average MRR@5 on the development set and placed third overall on the official test leaderboard with an average score of 0.7464.

The results show that the task benefits from multiple complementary stages. Sparse optimization provides lexical grounding, Harrier improves semantic candidate generation through scale, learned fusion handles the interaction between sparse and dense evidence, and Qwen3 reranking produces the strongest final ordering. The main practical limitation is resource usage: the best system depends on large GPU-backed models, while the fine-tuned BGE-M3 path remains more deployable when compute is constrained.

Future work should further assess whether longer reranker contexts justify their additional memory and latency. An investigation into how token length affects reranker performance also merits further study. Additional hard-negative mining may also help close part of the gap between smaller dense models and Harrier-scale retrieval.

Code Availability

The implementation of the system is available at https://github.com/mattiaskvist/check-that-clef-2026.

Acknowledgments

The authors thank the CLEF 2026 CheckThat! organizers for providing the task, data, and evaluation infrastructure. This paper and project were completed as part of a project in the course DD2477 - Search Engines and Information Retrieval Systems at KTH Royal Institute of Technology. We want to thank our Professor Johan Boye for the rewarding and interesting course, in addition to the iconic group name: Boyes Boys.

Declaration on Generative AI

During the preparation of this work, the authors used generative AI tools to assist with drafting, editing, and converting an earlier course report into the CEUR-WS working-notes format. The authors reviewed and edited the content as needed and take full responsibility for the publication's content.
Tools used for report writing and formatting: *GPT 5.5* in Codex for LaTex formatting and text creation, *Gemini 3.1 pro* in the Gemini app for report writing, reviewing and suggestions.

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, in: [2], 2026.

[2] E. S. Salido, A. Barrón-Cedeño, Alba García Seco de Herrera, S. MacAvaney, J. M. Struß (Eds.), CLEF 2026 Working Notes, CLEF 2026, Jena, Germany, 2026.

[3] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, F. Wei, Text embeddings by weakly-supervised contrastive pre-training, 2022. URL: https://arxiv.org/abs/2212.03533. doi:10.48550/arXiv.2212.03533. arXiv:2212.03533.

[4] 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, 2025. URL: https://arxiv.org/abs/2505.23250. doi:10.48550/arXiv.2505.23250. arXiv:2505.23250.

[5] S. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Foundations and Trends in Information Retrieval 4 (2009) 1–174. URL: https://doi.org/10.1561/1500000019. doi:10.1561/1500000019.

[6] A. Karpathy, Autoresearch: Autonomous ai research agents, https://github.com/karpathy/autoresearch, 2025. URL: https://github.com/karpathy/autoresearch, gitHub repository.

[7] GitHub, AutoResearch system skill for GitHub Copilot, Awesome Copilot Skills Library, 2025. URL: https://github.com/github/awesome-copilot/blob/main/skills/autoresearch/SKILL.md, accessed: 2026-07-06.

[8] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, Z. Liu, M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation, 2024. URL: https://arxiv.org/abs/2402.03216. doi:10.48550/arXiv.2402.03216. arXiv:2402.03216.

[9] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, 2021. URL: https://arxiv.org/abs/2106.09685. doi:10.48550/arXiv.2106.09685. arXiv:2106.09685.

[10] Microsoft, Harrier-OSS-v1: Multilingual text embedding models, https://huggingface.co/microsoft/harrier-oss-v1-27b, 2026. URL: https://huggingface.co/microsoft/harrier-oss-v1-27b, hugging Face model card.

[11] L. Breiman, Random forests, Machine Learning 45 (2001) 5–32. URL: https://doi.org/10.1023/A:1010933404324. doi:10.1023/A:1010933404324.

[12] NVIDIA, Llama Nemotron Reranking 1B v2, https://huggingface.co/nvidia/llama-nemotron-rerank-1b-v2, 2026. URL: https://huggingface.co/nvidia/llama-nemotron-rerank-1b-v2, hugging Face model card.

[13] F. Wang, Y. Li, H. Xiao, jina-reranker-v3: Last but not late interaction for listwise document reranking, 2025. URL: https://arxiv.org/abs/2509.25085. doi:10.48550/arXiv.2509.25085. arXiv:2509.25085.
[14] Y. Zhang, M. Li, D. Long, X. Zhang, H. Lin, B. Yang, P. Xie, A. Yang, D. Liu, J. Lin, F. Huang, J. Zhou, Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025. URL: https://arxiv.org/abs/2506.05176. doi:10.48550/arXiv.2506.05176. arXiv:2506.05176.

A. Random Forest Fusion

The random forest model for fusing the sparse and dense retrievers result into one ranked list, used the following setup:

Table 6
Random Forest settings

Parameter	Value
Estimators	100
Max depth	10
Min sample split	2

Hyperparameter tuning was conducted to yield the best setup. During training the top 500-1000 scores from sparse and dense retrievers was used in predicting the correct paper, depending on overlap between the two systems output. The model is available on hugging face: https://huggingface.co/boyes-boys-clef-2026/random-forest-fuser.

B. Autoresearch

B.1. Description of method

The optimization process began with a structured setup phase where the agent was initialized with a single-objective optimization task: maximizing the Mean Reciprocal Rank at Rank 5 (MRR@5) for the sparse retrieval class. To construct a closed-loop system, the agent was provided with:

1. The execution command (`uv run iterate_sparse_retrievers.py`) to run evaluations.
2. An extraction pattern (`FINAL_MICRO_AVG_MRR5: <SCORE>`) to automatically parse progress logs.
3. Strict operational constraints, restricting code edits exclusively to the file `iterate_sparse_retrievers.py` with an execution limit of under 10 minutes per experiment, up to a maximum budget of 50 total experiments.
4. Authorization to dynamically add system dependencies using the uv package manager to expand tokenization capabilities on the fly.

Operating under this declarative setup and sandboxed boundary, the agent automated a Git-based write-evaluate-revert cycle on a dedicated development branch (`autoresearch/apr06`). To maintain a clean and production-ready codebase, the agent implemented an automated branching hygiene strategy. When an experimental pipeline modification yielded a positive improvement in MRR@5, the agent committed the change and updated the running state as a kept baseline (status: `keep`). Conversely, if an experiment failed to surpass the previous best score (status: `discard`) or triggered an execution timeout (status: `crash`), the agent executed a programmatic rollback using a hard Git reset (`git reset –hard HEAD~1`).

This dual logging structure creates an important distinction between the production-facing version control history and the underlying scientific exploration trail. While the active Git repository contains only the clean, linear, and logical progression of successful changes, the agent's tracking register (`results.tsv`) captures the complete search space trajectory. Documenting these unsuccessful attempts is mathematically and methodologically valuable: it maps out the negative boundaries of the design space, prevents future researchers from duplicate search paths, and details key algorithmic bottlenecks. For example, the agent verified that utilizing BM25L (Exp. 5) led to severe performance degradation on this dataset (MRR@5 $= 0.0902$) compared to BM25Plus, and that higher-order n-grams like trigrams (Exp. 13) or non-adjacent skip-bigrams (Exp. 32) introduced vocabulary noise that reduced retrieval accuracy.

Over the course of the 50-experiment budget, the agent systematically evaluated a wide array of linguistic, structural, and mathematical parameters, including:

- **Preprocessing and Stopwords:** Transitioning from a minimal custom stopword list to NLTK's multilingual stopwords across English, German, and French, significantly reducing vocabulary noise.
- **Morphological Stemming:** Testing Porter, Snowball, and Lancaster stemmers, ultimately showing that the more aggressive Lancaster stemmer yielded the highest query-to-document match rate.
- **Phrase and Document Features:** Implementing word bigrams to capture local phrase context and boosting the weight of document titles (replicating the title field $3\times$ within the document text) to prioritize keyword matches in high-signal metadata zones.
- **Hyperparameter Tuning:** Transitioning the base scoring model from the default BM25Okapi to a tuned BM25Plus configuration, fine-tuning the document length normalization parameter to $b = 0.85$ and the term-frequency scaling parameter to $k_1 = 2.5$.

Through this automated optimization loop, the agent successfully drove the initial baseline performance from an MRR@5 of 0.3482 to a peak configuration scoring 0.4823, representing an overall relative improvement of $+38.5\%$. The complete, unedited chronological log of all 50 experiments (including discarded and crashed configurations) is provided in Appendix B.2.

B.2. Chronological Log of All Automated Experiments

The following table presents the complete sequence of modifications tested by the AutoResearch agent. This log contains both the successful changes that were committed to version control and the unsuccessful/failed iterations that were programmatically rolled back via Git resets during the optimization process.

Table 7: Complete optimization log of all 50 sparse retrieval experiments.

Exp. #	Commit Hash	Status	MRR@5	Description
0 (Base)	`1583c5e`	`baseline`	0.348228	Unmodified BM25Okapi baseline
1	`e5f6498`	`keep`	0.374257	Add multilingual stopword removal
2	`48b9713`	`keep`	0.424902	Add punctuation removal and min token length
3	`44eb516`	`keep`	0.425840	Switch from BM25Okapi to BM25Plus
4	`076fcac`	`discard`	0.411082	Include venue and authors in document text
5	`d8f2745`	`discard`	0.090166	Try BM25L instead of BM25Plus
6	`482bf5c`	`keep`	0.427045	Add English Snowball stemming
7	`1e66f73`	`keep`	0.432005	Expand stopwords list significantly
8	`17d67c7`	`keep`	0.436374	Increase BM25Plus k_1 parameter to 2.0
9	`bf76d3b`	`keep`	0.436391	Increase k_1 to 2.5
10	`23036e1`	`discard`	0.417041	Reduce b parameter to 0.5

Continued on next page

Table 7 – Continued from previous page

Exp. #	Commit Hash	Status	MRR@5	Description
11	`368009d`	keep	0.441768	Increase b parameter to 0.9
12	`fb871a4`	keep	0.453889	Add bigrams for phrase matching
13	`e19cba4`	discard	0.445831	Add trigrams for phrase matching
14	`e277742`	keep	0.461593	Boost title by repeating it twice
15	`9a5bf31`	keep	0.463990	Boost title by repeating it 3 times
16	`f342d01`	discard	0.462982	Increase k_1 to 3.0
17	`2983ab1`	crash	0.000000	Add character 3-grams for fuzzy matching (timeout)
18	`5e20d1c`	keep	0.465331	Remove minimum token length filter
19	`9812ad3`	discard	0.465331	Preserve numeric tokens without stemming (no change)
20	`31d041c`	keep	0.465699	Use PorterStemmer instead of SnowballStemmer
21	`6d6a23c`	discard	0.465699	Add delta = 0.5 to BM25Plus (no change)
22	`dd2f003`	discard	0.464166	Try BM25Okapi with tuned parameters
23	`66687c5`	discard	0.465288	Add scientific stopwords
24	`1089801`	keep	0.466330	Boost title $4\times$
25	`4fdab62`	keep	0.467585	Boost title $5\times$
26	`8cab50a`	discard	0.466480	Boost title $6\times$
27	`42b23ad`	discard	0.451996	Remove bigrams (simplify)
28	`0b82d9b`	discard	0.465942	Increase b to 0.95
29	`cfadab9`	discard	0.466380	Try k_1 = 2.0
30	`de8b7d2`	keep	0.471794	Use LancasterStemmer (more aggressive)
31	`18f33b5`	discard	0.471483	Add lemmatization before stemming
32	`5c7d031`	discard	0.468386	Add skip-bigrams (window = 2)
33	`c4c6432`	keep	0.472028	Try b = 0.85
34	`2f9d1ab`	keep	0.472064	Try b = 0.80
35	`7c89f2a`	discard	0.471276	Try b = 0.75
36	`7bf3201`	keep	0.481411	Use NLTK stopwords instead of custom list
37	`3b20d91`	discard	0.480620	Boost title $6\times$ with new settings
38	`d01f9e2`	keep	0.481691	Try b = 0.85 with NLTK stopwords
39	`8a10f92`	discard	0.481397	Try k_1 = 2.0 with new settings
40	`8f21bc9`	keep	0.481731	Try title $4\times$ with new settings
41	`a0c361b`	keep	0.482265	Try title $3\times$ (Peak configuration)
42	`2f10d9e`	discard	0.479792	Try title $2\times$
43	`d9e0231`	discard	0.481998	Try b = 0.90
44	`c0d12e9`	discard	0.472538	Try SnowballStemmer with new stopwords
45	`3fa829d`	discard	0.482182	Try k_1 = 1.8
46	`b82dc10`	discard	0.481581	Try k_1 = 3.0
47	`a182f3a`	discard	0.452427	Use only English stopwords
48	`6cd01f3`	discard	0.481818	Add positional markers for first 3 words
49	`ef23bc1`	discard	0.481951	Try k_1 = 2.3
50	`298ab3c`	discard	0.482232	Try b = 0.82