Varytrieval at CheckThat! 2026: Combining Rule-Based String Matching and Dense Retrieval for Scientific Claim Source Retrieval

Alica Hövelmeyer[1]

[1]Heinrich Heine University Düsseldorf, Universitätsstraße 1, 40225 Düsseldorf, Germany

Abstract

This paper presents *varytrieval*, a retrieval system developed for Task 1 of the CheckThat! 2026 Lab at CLEF, which addresses source retrieval for scientific web claims. Given a social media post in English, German, or French, the task is to identify the scientific paper implicitly referenced by the post from a shared candidate collection. The system combines rule-based string matching with dense retrieval using a *Qwen3-Embedding-0.6B* sentence encoder fine-tuned with hard negatives. The rule-based components are used when posts contain explicit signals, such as title fragments, author names, journal names, quotations, or distinctive lexical and entity-based cues. Cases not covered by these components are handled by the dense retriever. For German and French, queries are translated into English before retrieval, and a cross-encoder is applied as a final reranking step. The system achieves an MRR@5 of 0.6914 on the English split, suggesting that hard-negative fine-tuning and metadata-aware string matching are effective for scientific source retrieval.

Keywords

scientific claim retrieval, source retrieval, fact-checking, dense retrieval, string matching, multilingual retrieval

1. Introduction

The *CLEF* conference's *CheckThat! Lab* is focused on "the spreading of mis- and disinformation across multiple platforms and languages in online discourses." One objective within this context is the retrieval of sources for claims made on the internet. This process is challenging because scientific sources are often referenced outside formal academic contexts. Even when social media posts refer to specific scientific papers, they may do so only implicitly, for example, by mentioning a title fragment, an author, a journal, a quotation, or a vague description of the paper's findings. The first task of this year's lab edition deals precisely with this problem. The official task description reads as follows:

Given a social media post that contains a scientific claim and an implicit reference to a scientific paper (mentions it without a URL), retrieve the mentioned paper from a pool of candidate papers. [1]

The respective social media posts possess varying degrees of formality. Some cite papers in an almost scientific manner, while others contain only vague assertions. These varying modes of reference call for complementary retrieval strategies, which are addressed by the *varytrieval* system. A fine-tuned sentence encoder accounts for the largest part of the system's retrieval performance, capturing the semantic relationship between post and paper. Beyond semantic similarity, however, many posts contain direct references to metadata such as paper titles, authors, journals, citations, or quoted text. Therefore, several rule-based components are integrated into the ranking process. These components form a significant part of the system and improve the overall retrieval performance by about 1.3%.

In this year's edition, the task is multilingual for the first time, covering English, German, and French queries. For German and French, the *varytrieval* system translates queries into English before retrieval, and a general-purpose cross-encoder is used as the final reranking stage for these language splits.

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

alica.hoevelmeyer@hhu.de (A. Hövelmeyer)

2. Related Work

The *CheckThat! Lab* was first introduced in 2018 as part of the *Conference and Labs of the Evaluation Forum (CLEF)*, with the aim of promoting the development of technologies to counter the spread of mis- and disinformation across different platforms and languages [2]. Over the course of its editions, the lab has addressed central components of the fact-checking pipeline, such as check-worthiness detection, evidence retrieval, and claim verification [3].

The specific subtask of source retrieval for scientific web claims was introduced in 2025 as part of a broader task on scientific web discourse detection and source retrieval. Its relevance stems from the increasing spread of science communication on social media [4]. In the 2026 edition, the task was elevated to a standalone task. Moreover, the scope of the original task was extended by introducing a multilingual setting, covering English, German, and French [5].

In the 2025 edition of the retrieval task, 30 teams participated, with seven submitting system description papers. Most approaches applied an initial candidate retrieval step followed by re-ranking. *AIRwaves* [6] ranked second. For retrieval, a fine-tuned *E5-large* dual encoder with hard negative mining was used. Re-ranking was done with a *SciBERT* cross-encoder. The system reached an MRR@5 of 0.6828 on the test set. *Deep Retrieval* [7] ranked third. For retrieval, they combined *BM25* with a *FAISS*-based dense search using a fine-tuned *INF-Retriever-v1* model. Re-ranking was performed by an LLM-based cross-encoder, achieving an MRR@5 of 0.6643. *ATOM* [8] combined *BM25* and *GTR-T5-Large* for retrieval, and applied *MonoT5-base-med-msmarco* for pointwise re-ranking, additionally evaluating listwise re-ranking strategies. *SeRRa* [9] ranked eighth and used a three-step pipeline: *Sentence-BERT* for dense retrieval of the top 200 candidates, a fine-tuned *SciBERT* model for binary classification re-ranking, and a final pairwise comparison model over the top 10 results. *SCIRE* [10] used a dense retriever to produce the top 20 candidates, which were then re-ranked by a cross-encoder. The system additionally integrated a scientific discourse detection step using ensemble *DeBERTa* models.

Two teams took different directions. *Claim2Source* [11] applied zero-shot style transfer via an LLM to rewrite informal tweet queries into formal scientific language before retrieval, testing this approach across 15 retrieval systems, including sparse, dense, and hybrid variants, with *GritLM-7B* performing best. *DS@GT* [12] ranked 16th and used *BM25-Pytorch* for retrieval and *T5* for re-ranking, additionally exploring six data augmentation techniques and fine-tuning a bi-encoder, reaching an MRR@5 of 0.58.

The 2025 systems reflect a broad range of information retrieval methods. A still widely applied baseline is *BM25* [13], a sparse retrieval model that ranks documents based on term frequency and inverse document frequency. It is often used as a first-stage candidate generator within hybrid pipelines, providing an initial candidate set for subsequent neural re-ranking stages. However, in many recent systems, the core retrieval step relies on semantic similarity estimation using bi-encoder models. Reimers and Gurevych popularized this approach for sentence-level semantic similarity with *Sentence-BERT*, a siamese network structure that produces semantically meaningful sentence embeddings [14]. More recent approaches build on large language model backbones rather than encoder-only architectures. The *Qwen3-Embedding* model [15] used in this work follows this direction, being built on top of the *Qwen3* foundation model and trained through a multi-stage pipeline of unsupervised pre-training and supervised fine-tuning. To further improve ranking precision, cross-encoder re-rankers are applied to a reduced candidate set produced by the retrieval stage. Unlike bi-encoders, cross-encoders jointly encode query and document, enabling fine-grained interaction between the two (see [16, 17]). The code to recreate our results is available here: https://github.com/Alihoe/varytrieval.

3. Task and Data

The task consists of ranking, for each query drawn from a dataset of queries, the most relevant document from a given document collection. The queries are derived from social media posts published on *X*. In this year's edition, the queries are provided in three languages for the first time: English, German, and French. For the English training split, the training data from the previous year's edition is reused, while

all remaining splits are newly annotated [5]. Table 1 contains the sizes of the data splits.

Table 1
Number of queries per data split.

Language	Train	Dev	Test
English	14,977	3,905	6,076
German	1,460	386	876
French	2,807	702	1,220

The collection contains 10,000 documents and is shared across all three languages. It consists of scientific studies, represented by the fields *title*, *abstract*, *venue*, and *authors*. The documents used in the original task were collected from the *CORD-19* corpus ([3], [4]). The collection thus contains papers related to *COVID-19* research, but there are also other topics in science and medicine covered. See Figure 1 for an example.

> **query:** *We perceive the world with molecular machines that evolved across the tree of life to solve varied sensory problems in various species. This fascinating new paper shows how the umami taste sense in songbird ancestors evolved into a sweet tooth, er, beak.*
> **gold document title:** *Early origin of sweet perception in the songbird radiation*

Figure 1: A sample instance from the 2026 English dev set.

The task is evaluated using *MRR@5*, a metric that assigns a score equal to the reciprocal rank of the first relevant item, considering only the top five retrieved documents [18].

4. Methodology

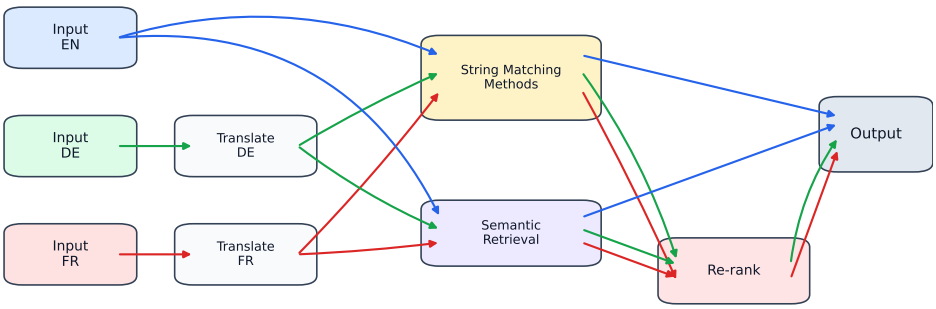

Figure 2: Overview of the *varytrieval* pipeline.

The system consists of four main steps. First, the German and French language splits are translated into English. The second step aims to find query-document matches that are correct with high probability, using string matching methods adapted to the dataset. Similar to the approaches presented in the related work section, this is then followed by a retrieval step using a fine-tuned sentence encoder. This is applied to all language splits. Finally, a general-purpose cross-encoder for re-ranking is applied to the French and German language splits.

4.1. Pre-Processing

We experimented with various pre-processing methods. First, we applied normalization in the form of lowercasing and whitespace normalization to both the queries and the documents. As a baseline for evaluating these methods, we used the standalone MRR@5 performance of sentence encoder *Qwen3-Embedding-0.6B*. Interestingly, unlike with last year's data, the normalization yielded no improvement. In particular, normalizing the queries worsened the results (see Table 2).

Table 2
MRR@5 using the *Qwen3-Embedding-0.6B* model (not fine-tuned) as a standalone component with and without normalization.

Datasplit	Raw	Query Norm	Doc Norm	Query & Doc Norm
2026 EN DEV	**0.5846**	0.5793	0.5824	0.5818
2026 DE DEV	**0.4543**	0.4430	0.4452	0.4373
2026 FR DEV	**0.5858**	0.5816	0.5854	0.5786
2025 EN DEV	0.6638	0.6657	**0.6672**	0.6652

As document fields, we included a combination of title and abstract. These are the fields that contain most of the semantic information and also less noise than some of the author and venue fields. The decision to only use these fields was further supported by the fact that the string matching methods are specifically designed to capture matches based on authors and venues.

The German and French datasets are translated to English using a local translation setup via *Ollama* [19]. After also experimenting with *llama3.2:3b* [20], we use *translategemma:4b* [21] as the translation model. This clearly improves the overall semantic retrieval results. As illustrated in Tables 3 and 4, the performance gain is dependent on the translation model used. This suggests that further improvements may be possible with stronger or more specialized translation models. For more details about the translation setup, see section 8.1.

Table 3
MRR@5 of the German dev set using the *Qwen3-Embedding-0.6B* model as a standalone component with or without translation.

Dataset	No Translation	llama3.2:3b	translategemma:4b
With Fine-Tuning	0.5258	0.5263	**0.5559**
Without Fine-Tuning	0.4541	0.4257	**0.4543**

Table 4
MRR@5 of the French dev set using the *Qwen3-Embedding-0.6B* model as a standalone component with or without translation.

Dataset	No Translation	llama3.2:3b	translategemma:4b
With Fine-Tuning	0.6322	0.6567	**0.6676**
Without Fine-Tuning	0.5511	0.5617	**0.5858**

4.2. String Matching Methods

As the first stage of the pipeline, we use string-matching methods. These are intended to identify cases in which the prediction is straightforward and unambiguous because the relevant document data is cited almost verbatim. The different rule-based methods exploit different types of references and either produce exact matches or generate a high-quality set of candidate matches. The components are applied as a sequential filtering stage. For each query, the first string-based method that returns a non-empty result claims the query. Other string-based methods are not applied to it. If the method returns several

Table 5
MRR@5 on 2026 EN DEV using the different string matching methods with a *Qwen3-Embedding-0.6B* model as the fallback. The abbreviations refer to the methods in their order of presentation in the following sections (string similarity detector, token similarity detector, author detector, title matcher, journal detector, quote detector, named entity detector).

	dense only	**s**	**s+to**	**s+to+a**	**s+to+a+ti**	**s+to+a+ti+jo**	**s+to+a+ti+jo+q**	**all**
With Fine-Tuning	0.6798	0.6797	0.6802	0.6827	0.6850	0.6887	0.6885	**0.6887**
Without Fine-Tuning	0.5846	0.5850	0.5864	0.5924	0.5969	0.6022	0.6023	**0.6025**

Table 6
Performance comparison between dense retrieval and the string-based pipeline on the subset of development queries matched by at least one string-matching component.

System	MRR@5	Correct@1	Correct@5
Dense retrieval	0.792	0.762	0.833
String/full pipeline	0.868	0.823	0.926
Absolute difference	+0.076	+0.061	+0.093

candidate documents, these candidates are ordered using the same semantic retriever that is used as the fallback for queries not handled by the string-based stage. If no string-based method returns a result, the full dense retrieval stage is applied. Most string-based components return candidate lists rather than final single-document predictions. The Token Similarity Detector, Author Detector, Title Matcher, Journal Detector, and Quote Detector produce candidates that are ordered with the semantic retriever, while the Named Entity Detector ranks its candidates directly by shared linked entities.

Together, these methods improve results by about 1.3% from an MRR@5 of 0.6798 to 0.6887 using the fine-tuned sentence encoder as a fallback. However, using the methods and the sentence encoder without fine-tuning as a fallback improves results by over 3% (Table 5). This suggests that the fine-tuned encoder learns some of the lexical similarities targeted by the string-matching methods, making their additional contribution smaller, but not irrelevant once fine-tuning is applied. On the subset of queries actually matched by the string-based stage, the pipeline improves over dense retrieval alone by 0.0759 MRR@5, as shown in Table 6.

Table 7 shows how many queries are handled by the different components. They are presented in the same order as they are employed in the pipeline.

4.2.1. String Similarity Detector

The first string matching component in the pipeline is a general string similarity detector. It is used for cases in which the query still has a clear surface-level overlap with the title or abstract of the target document. First, the detector retrieves a small set of possible documents by using rare tokens and character-level n-grams. These candidates are then compared with the query by using token overlap, character overlap, and sequence-based similarity measures. A match is only accepted if the query is also supported by a local span in the document. The detector returns a prediction only when one document is clearly stronger than the other candidates. In cases of weak evidence, short queries, or close competing matches, it abstains from classification. For more details, see Appendix 8.2.

4.2.2. Token Similarity Detector

The token similarity detector handles cases in which the query contains several distinctive rare tokens from the target document. Common and less informative tokens are removed. The remaining query tokens are then ranked by their IDF weight, and the rarest tokens are used as the query signature. A

document is only treated as a possible match if it contains this signature. It must also share enough additional informative tokens with the query. For more details, see Appendix 8.3.

4.2.3. Author Detector

The author detector is designed for queries that explicitly mention one or more authors of the target paper. For each paper, several normalized forms of the author names are stored. These include full names, surnames, and forms based on initials. When a query is processed, the detector searches for these name forms in the query text. It also checks the surrounding context, because not every name occurrence is necessarily an author reference. Citation-style wording, author-related cue words, and phrases such as *et al.* are treated as supporting evidence. At the same time, misleading matches are filtered out. This includes names that occur as institutions, social handles, group names, or common words. The exact cues are adapted to the respective language. The remaining author evidence is linked to candidate papers. The detector is mainly used to produce a strong candidate set for later reranking, not to return definite matches on its own. More details are given in Appendix 8.4.

4.2.4. Title Matcher

The title matcher covers queries in which the title of a paper is mentioned directly or almost directly. Document titles and queries are first normalized into comparable token sequences. The matcher then searches the query for exact title spans. Very short or uninformative title matches are ignored. This avoids that common phrases are treated as reliable paper references. If a title match is found, the component creates candidates from the matched title and ranks them according to the strength of this exact match. More details are given in Appendix 8.5.

4.2.5. Journal Detector

The journal detector is designed for queries that explicitly mention the journal or publication venue of the target paper. Journal names and common aliases are first normalized and stored as token sequences. The detector then searches for these forms in the query. A journal match is only used if the surrounding context makes a publication venue likely. This includes phrases such as *published in* or *appeared in*. Ambiguous cases are filtered out, especially when a journal name could also describe a topic or discipline, for example *health*, *medicine*, or *science*. The remaining journal matches are mapped to candidate papers. They can also be expanded through journal-family relations. The candidates are then ranked by using the journal evidence together with the title and abstract overlap. If the journal evidence is too weak, or if it points to several unrelated journal families, the detector abstains. Like the author detector, this method mainly produces a strong candidate set for later reranking and not definite matches on its own. More details can be found in Appendix 8.6.

4.2.6. Quote Detector

The quote detector is used for queries that contain a direct quotation from the target paper. In contrast to the other string-based components, it uses the full paper texts and not only metadata such as titles, authors, or journals. The detector first extracts quoted spans from the query and normalizes them. It then checks whether the same text occurs exactly in a document title or in the full text. Very short quotes are ignored, because they are often too general to identify a paper in a reliable way. Only exact normalized quote matches are used as evidence. If a quote matches only a small number of documents, these documents are returned as candidates. Closely related records with the same or very similar titles can also be added. If the quote is not found, or if it matches too many different documents, the detector abstains. For more details, see Appendix 8.7.

Table 7
Number of queries of the 2026 English dev set handled by the separate components.

Component	Predicted	Correct@5	Incorrect@5
Token Similarity Detector	73	72	1
Author Detector	154	144	10
Title Matcher	104	104	0
Journal Detector	110	87	23
Quote Detector	7	7	0
Named Entity Detector	15	15	0
Detector/Matcher Total	463	429	34
Fallback	3442	2562	880
Total	3905	2991	914

4.2.7. Named Entity Detector

The named entity detector is a slightly different component. It is used for queries that refer to a paper through several distinctive entities, instead of through exact title, author, journal, or quote matches. Instead of comparing the surface text directly, the detector links entities in the query and in the documents to Wikidata identifiers. It then compares these linked entity sets. A prediction is only returned if enough linked entities are found in the query, if a document shares several of them, and if at least one shared entity is rare in the collection. This helps the detector to focus on specific combinations of entities and not only on broad or common topics. If too few entities are found, if many documents share the same entities, or if there is no clearly strongest candidate, the detector abstains. More details are given in Appendix 8.8.

4.3. Semantic Retrieval

After the rule-based detectors, the pipeline uses a semantic retrieval step for all remaining cases. This stage uses a fine-tuned *Qwen3-Embedding-0.6B* [15] sentence encoder as a dense retriever. We experimented with different retrievers on the evaluation data from the 2025 edition and achieved the best results with this model family (see Table 8). Although the larger *Qwen3-Embedding-8B* achieved better results without fine-tuning, the smaller *Qwen3-Embedding-0.6B* model outperformed its larger counterpart after fine-tuning. We decided to use the smaller model for fine-tuning, rather than also fine-tuning the larger model, to be able to experiment and to not waste too many resources.

Table 8
Sentence Encoder Model Comparison on the 2025 Test Set using MRR@5.

Model	2025 EN Test
Qwen3-0.6B	0.5624
Qwen3-0.6B finetuned on 2025 data	**0.6346**
Qwen3-8B (large)	0.6131
E5-large-v2	0.5067
Multilingual-e5-large-instruct	0.5279
Bge-large-en-v1.5	0.4992
Bge-multilingual-gemma2 (large)	0.5128
NV-Embed-v2 (large)	0.5998

The model is fine-tuned on the training data from the 2025 edition. This yielded the best results among all tested configurations. We experimented with adapter-based fine-tuning and full fine-tuning strategies (see Table 9) and decided to use full fine-tuning. The *Qwen* encoder is trained with *MultipleNegativesRankingLoss* in a hard-negative setup. Each training instance contains a query, its gold

document, and several mined non-relevant documents, encouraging the model to score the gold document above difficult distractors. In the final configuration, we use ten hard negatives. These negatives are constructed from three sources: two negatives come from *BM25* retrieval, four negatives come from the base *Qwen3-Embedding-0.6B* encoder and four negatives come from a previously fine-tuned *Qwen* retriever. The inclusion of *BM25*-based hard negatives follows the approach of team *AIRwaves* in last year's edition [6]. To complement these lexical hard negatives, we also include semantic hard negatives to better reflect challenging cases for the retriever being fine-tuned. Duplicate documents are removed, and if one source does not provide enough unique negatives, the remaining slots are filled from the other sources. Fine-tuning details are provided in Appendix 8.9.

Table 9
MRR@5 scores for different *Qwen3-0.6B* fine-tuning setups on the 2026 English development set.

Configuration	2026 EN Dev
Full fine-tuning on 2025 data, ten negatives	**0.6798**
Full fine-tuning on all 2025 data, ten negatives	0.6574
Full fine-tuning on all training data, ten negatives	0.6348
Full fine-tuning on 2026 data, ten negatives	0.5846
Adapter fine-tuning on 2025 data, five negatives	0.6208
Adapter fine-tuning on 2025 data, ten negatives	0.6233
Adapter fine-tuning on 2026 EN data, five negatives	0.6106
Adapter fine-tuning on 2026 data, ten negatives	0.6190

This component accounts for the strongest performance gain in our system, increasing MRR@5 performance on the English dev set from 0.5846 to 0.6798.

4.4. Cross-Encoder Reranking

As the final stage of the pipeline, a cross-encoder reranking step is applied to the German and French datasplits using the *bge-reranker-v2-gemma* [22, 23] model without fine-tuning. Unlike the dense retriever, the cross-encoder model scores each query-document pair jointly. This takes more computational effort, and thus only a small candidate pool from the previous retrieval stage is reranked. We experimented with different cross-encoders, including *mxbai-rerank-large-v2* and *bge-reranker-base* [24] with and without fine-tuning, because they were successful in the previous task edition [7, 8]. Although these cross-encoder models improved results when applied to candidates from the dense retriever before fine-tuning, they worsened results when applied to candidates from the fine-tuned encoder. For the English dev set, we could not achieve any improvement by applying these models. This might be due to the fact that we had to let the large *bge-reranker-v2-gemma* model run with reduced precision to handle the amount of compute. Furthermore, it suggests that the sophisticated fine-tuning of the sentence encoder produces a vector space that is semantically rich enough to make pairwise document comparison unnecessary. For the other language splits, we achieved some improvements using *bge-reranker-v2-gemma* with different numbers of re-ranking candidates (see Table 10). In the final setup, the reranking depth is 5 for German and 10 for French.

Table 10
MRR@5 scores on the dev sets for the tested re-ranking candidate pool settings.

Language	No Rerank	5	5+5 BM25	10
EN	**0.6887**	0.6778	0.6745	0.6755
DE	0.5592	**0.5693**	0.5626	0.5651
FR	0.6750	0.6941	0.7032	**0.7110**

5. Results

Overall, the *varytrieval* system achieved a moderate result on the evaluation leaderboard. It finished 12th out of 35 teams, with an overall average score of 0.6388. While the combined multilingual score was below the strongest systems, the English subsystem performed comparatively well. In the English evaluation, *varytrieval* reached a score of 0.6914. This is competitive with, and in some cases higher than, the English scores of several teams that ranked higher overall, for example the systems at ranks 6, 8, 9, 10, and 11 (see Table 11). This indicates that the German and French subsystems still need further adjustments, but the overall retrieval strategy works reasonably well for English web claims.

Table 11
System performance comparison on the competition leaderboard.

Rank	System	Avg MRR@5	DE MRR@5	EN MRR@5	FR MRR@5
1	claim2source	0.7628	0.7153	0.7819	0.7912
2	team-outsider	0.7580	0.7228	0.7802	0.7709
3	mattiaskvist	0.7464	0.7070	0.7571	0.7752
4	CheckMate	0.7194	0.6861	0.7343	0.7378
5	seupd2526-retrix	0.7108	0.6806	0.7117	0.7400
6	mever_checkthat	0.6839	0.6615	0.6829	0.7074
7	farnaz	0.6819	0.6455	0.6980	0.7021
8	amr_abdelsamea	0.6597	0.6228	0.6650	0.6912
9	thechuongchu	0.6489	0.6319	0.6528	0.6620
10	awakened	0.6405	0.6130	0.6443	0.6642
11	mg_ct2026	0.6394	0.5971	0.6382	0.6828
12	**varytrieval**	**0.6388**	**0.5815**	**0.6914**	**0.6436**
13	seupd2526-fella	0.5961	0.5638	0.6062	0.6183
14	Galaxy	0.5959	0.5603	0.6020	0.6255
15	BoPC	0.5940	0.5620	0.5930	0.6270
16	seupd-cobalt	0.5746	0.5407	0.5775	0.6057
17	teamxclef2026	0.5677	0.5208	0.5973	0.5850
18	bhargab_jb	0.5565	0.5014	0.5961	0.5719
19	joseph	0.5501	0.4899	0.5739	0.5864
20	huesos	0.5435	0.4870	0.5697	0.5738
21	meth3group2	0.5429	0.4989	0.5652	0.5647
22	valentin-tian	0.5332	0.4915	0.5422	0.5659
23	clara	0.5137	0.4571	0.5469	0.5371
24	sagasu	0.5098	0.4379	0.5606	0.5309
25	cam_le	0.5062	0.4493	0.5248	0.5445
26	seupd2526-dell	0.5000	0.4461	0.5302	0.5238
27	stefanoriato	0.4972	0.4657	0.4858	0.5400
28	mailuefterl	0.4926	0.4157	0.5570	0.5051
29	fmi-bionlp-aac	0.4901	0.4354	0.4879	0.5470
30	CheckMates	0.4891	0.4119	0.5381	0.5174
31	rpapusoi	0.4870	0.4072	0.5548	0.4992
32	Infragylph	0.4825	0.4319	0.5056	0.5099
33	mary_toapanta	0.4328	0.3734	0.4943	0.4306
34	lamungu	0.2401	0.1286	0.4535	0.1382
35	elineevje	0.2346	0.0854	0.5348	0.0835
36	ct_tutorial_acc	-1.0000	0.3794	-1.0000	-1.0000
37	mohotarema	-1.0000	-1.0000	0.6439	-1.0000

6. Conclusion

This paper presents *varytrieval*, a retrieval system for matching informal scientific claims from social media posts to the scientific papers they implicitly reference. The system combines rule-based string matching, dense semantic retrieval, and cross-encoder reranking. Its strongest component is the fine-tuned *Qwen3-Embedding-0.6B* sentence encoder, which is trained with hard negatives. With this training setup, the model learns to separate the correct paper from documents which are lexically or semantically very close. This makes the encoder especially useful for this task.

The string matching components improve the performance most clearly when the encoder is not fine-tuned. After fine-tuning, their additional contribution becomes smaller. This suggests that hard-negative training already helps the dense retriever to capture some forms of surface-level overlap between posts and papers. Still, the string-based methods remain useful. They provide high-precision predictions for cases with explicit references to titles, authors, journals, quotations, or distinctive tokens. They also make the system easier to interpret. Their value is therefore not only performance-based, but also explanatory.

The final results show that the system works especially well for English. The overall multilingual leaderboard score is limited by a weaker German and French performance, but the English subsystem achieves a competitive result with a relatively simple and inexpensive design. For German and French, the system relies on translation into English. This helps dense retrieval, but it also adds another possible source of error.

Future work should therefore focus on adapting the rule-based components more clearly to multilingual data, improving the translation pipeline, or fine-tuning retrievers directly on language-specific examples. Overall, the results indicate that hard-negative fine-tuning is very effective for improving retrieval performance when candidate documents are semantically close. At the same time, string similarity methods remain useful as precise, interpretable, and computationally cheap complements to neural retrieval.

7. Declaration on Generative AI

During the preparation of this work, the author used Grammarly for spelling checks, ChatGPT (GPT-5.5 Instant) for translation, writing style improvement and citation management, and Gemini (Gemini 3.5 Flash) for formatting assistance. After using these tools, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] J. M. Struß, S. Schellhammer, S. Dietze, V. V., V. Setty, T. Chakraborty, P. Nakov, A. Anand, P. Chungkham, S. Hafid, D. Sahnan, K. Todorov, The clef-2026 checkthat! lab: Advancing multilingual fact-checking, in: R. Campos, A. Jatowt, Y. Lan, M. Aliannejadi, C. Bauer, S. MacAvaney, A. Anand, Z. Ren, S. Verberne, N. Bai, M. Mansoury (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2026, pp. 325–335.

[2] P. Atanasova, A. Barrón-Cedeño, T. Elsayed, R. Suwaileh, W. Zaghouani, S. Kyuchukov, G. D. S. Martino, P. Nakov, Overview of the CLEF-2018 checkthat! lab on automatic identification and verification of political claims. task 1: Check-worthiness, CoRR abs/1808.05542 (2018). URL: http://arxiv.org/abs/1808.05542. arXiv:1808.05542.

[3] F. Alam, J. M. Struß, T. Chakraborty, S. Dietze, S. Hafid, K. Korre, A. Muti, P. Nakov, F. Ruggeri, S. Schellhammer, V. Setty, M. Sundriyal, K. Todorov, V. V, The clef-2025 checkthat! lab: Subjectivity, fact-checking, claim normalization, and retrieval, 2025. URL: https://arxiv.org/abs/2503.14828. arXiv:2503.14828.

[4] 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: G. Faggioli,

N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_54.pdf.

[5] J. M. Struß, S. Schellhammer, S. Dietze, V. V, V. Setty, T. Chakraborty, P. Nakov, A. Anand, P. Chungkham, S. Hafid, D. Sahnan, K. Todorov, The clef-2026 checkthat! lab: Advancing multilingual fact-checking, 2026. URL: https://arxiv.org/abs/2602.09516. arXiv:2602.09516.

[6] C. Ashbaugh, L. Baumgärtner, T. Greß, N. Sidorov, D. Werner, AIRwaves at CheckThat! 2025: Retrieving scientific sources for implicit claims on social media with dual encoders and neural re-ranking, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_63.pdf.

[7] 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, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_89.pdf.

[8] M. Staudinger, A. El-Ebshihy, W. Kusa, F. Piroi, A. Hanbury, ATOM at CheckThat! 2025: Retrieve the implicit – scientific evidence retrieval, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_98.pdf.

[9] G. A. Marchetti, G. Rocha, H. Lopes Cardoso, Team SeRRa at CheckThat! CLEF 2025: Sequential re-ranking in a scientific claim source retrieval pipeline, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_81.pdf.

[10] P. M. Thapliyal, R. S. Chavan, S. Samridh, C. Zuo, R. Banerjee, SCIRE at CheckThat! 2025: Bridging social media, scientific discourse, and scientific literature, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_99.pdf.

[11] T. Schreieder, M. Färber, Claim2Source at CheckThat! 2025: Zero-shot style transfer for scientific claim-source retrieval, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://ceur-ws.org/Vol-4038/paper_94.pdf.

[12] J. Schofield, M. Heil, et al., DS@GT at CheckThat! 2025: Exploring retrieval and reranking pipelines for scientific claim source retrieval on social media discourse, in: G. Faggioli, N. Ferro, P. Rosso, D. Spina (Eds.), Working Notes of CLEF 2025 – Conference and Labs of the Evaluation Forum, volume 4038 of *CEUR Workshop Proceedings*, CEUR-WS.org, Madrid, Spain, 2025. URL: https://arxiv.org/abs/2507.06563.

[13] S. Robertson, H. Zaragoza, The probabilistic relevance framework: Bm25 and beyond, Foundations and Trends in Information Retrieval 3 (2009) 333–389. doi:10.1561/1500000019.

[14] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, CoRR abs/1908.10084 (2019). URL: http://arxiv.org/abs/1908.10084. arXiv:1908.10084.

[15] 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. arXiv:2506.05176.

[16] R. Nogueira, K. Cho, Passage re-ranking with BERT, CoRR abs/1901.04085 (2019). URL: http://arxiv.org/abs/1901.04085. arXiv:1901.04085.

[17] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, J.-Y. Nie, C-pack: Packed resources for general chinese embeddings, 2024. URL: https://arxiv.org/abs/2309.07597. arXiv:2309.07597.

[18] E. M. Voorhees, D. M. Tice, The TREC-8 question answering track, in: Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00), European Language Resources Association (ELRA), Athens, Greece, 2000. URL: https://aclanthology.org/L00-1018/.

[19] Ollama, Ollama, 2026. URL: https://github.com/ollama/ollama, local platform for running open models.
[20] Ollama, llama3.2:3b, 2024. URL: https://ollama.com/library/llama3.2:3b, ollama model library entry for Meta Llama 3.2 3B.
[21] M. Finkelstein, I. Caswell, T. Domhan, J.-T. Peter, J. Juraska, P. Riley, D. Deutsch, C. Dilanni, C. Cherry, E. Briakou, E. Nielsen, J. Luo, K. Black, R. Mullins, S. Agrawal, W. Xu, E. Kats, S. Jaskiewicz, M. Freitag, D. Vilar, Translategemma technical report, 2026. URL: https://arxiv.org/abs/2601.09012. arXiv:2601.09012.
[22] C. Li, Z. Liu, S. Xiao, Y. Shao, Making large language models a better foundation for dense retrieval, 2023. arXiv:2312.15503.
[23] 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, 2024. arXiv:2402.03216.
[24] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, C-pack: Packaged resources to advance general chinese embedding, 2023. arXiv:2309.07597.
[25] Ollama, TranslateGemma 4B, https://ollama.com/library/translategemma:4b, 2025. Accessed: 2026-05-18.
[26] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spacy: Industrial-strength natural language processing in python (2020). doi:10.5281/zenodo.1212303.
[27] Explosion, en_core_web_sm: English pipeline for spacy, https://spacy.io/models/en, 2026. Accessed: 2026-05-20.
[28] E. Gerber, spaCy Entity Linker: Linked entity pipeline for spacy, https://github.com/egerber/spacy-entity-linker, 2023. Version 1.0.3; accessed: 2026-05-20.

8. Appendix

8.1. Translation Setup

We used the following prompt for translation with the *translategemma:4b* model, as recommended in the respective *Ollama* prompt guide [25]:

```
You are a professional French/German to English translator.
Your goal is to accurately convey the meaning and nuances of the original French/German text
while adhering to English grammar and vocabulary.
Produce only the English translation, without any additional explanations or commentary.
Please translate the following French/German text into English:
```

An output is rejected for the following reasons:

- empty output
- multi-line output
- refusal-like output (containing terms such as *i can't*, *i cannot*, *i cant*, *i'm unable*, *i am unable*, *cannot assist*, *can't assist*, *cannot help*, *can't help*, *cannot rewrite*, *can't rewrite*, *can't fulfill*, *cannot fulfill*, *can i help you with something else*, *is there anything else i can help*, *misinformation*, *harmful*, *dangerous activities*, *medical advice*, *policy*, *safety*)
- framing-output (containing terms such as *here's a rewritten*, *here is a rewritten*, *rewritten query:*, *rewrite:*, *output:*, *neutral rewrite:*, *here's an attempt*, *here is an attempt*, *please note*, *would you like me to*, *in a more neutral tone*, *i can rephrase*, *i can rewrite*, *i can help rephrase*, *the rewritten query*, *this rewritten query*, *here's a rewritten version*, *here is a rewritten version*, *here is the translation*, *here is your translation*, *hier ist die übersetzung*, *translation:*, *translated text:*, *translated:*, *here's the translation*, *here's your translation*, *the translation is*, *this is the translation*, *english translation:*,

german translation:, *french translation:*, *spanish translation:*, *italian translation:*, *übersetzung:*, *hier ist die übersetzung*)

After rejection, up to two new translation attempts are made.

8.2. String Similarity Detector Details

For queries and documents, the same preprocessing is applied. The text is first lowercased and normalized with Unicode *NFKC* normalization. Afterwards, it is tokenized and put together again with one blank space between the tokens. Leading and trailing whitespace is removed, and several consecutive whitespace characters are reduced to one. For the character-based comparison, character trigrams are built on the normalized text after all spaces have been removed.

Before candidates are retrieved, queries which are too short or not specific enough are excluded. The following rules are used:

- A query has to contain at least five normalized tokens and at least 24 normalized characters.
- If these conditions are fulfilled, an exact-match check is done first. If the normalized query is identical to the complete text of exactly one document, this document is returned directly.
- If the same normalized text belongs to more than one document, the detector gives no prediction for this case.

For all queries which are not solved by the exact-match rule, candidates are retrieved with the help of rare query tokens and rare character trigrams. This keeps the number of possible documents limited, while still keeping documents which are likely to match. In this configuration, the following limits are used:

- At most seven query tokens and eight character trigrams are used as seeds for the retrieval.
- Tokens which occur in more than 8% of the documents are not used. Character trigrams which occur in more than 4% of the documents are also ignored.
- The candidate set is limited to 300 documents.

The candidates are then reduced and ordered before the full scoring is calculated:

- Candidates with almost no lexical overlap are removed. For this, a document has to share at least one token with the query, or at least three character trigrams.
- The remaining candidates are ordered by token overlap and character-trigram overlap. From this ordering, only the best 60 candidates are kept for the full scoring.

The full score is calculated from several similarity measures. It includes soft query coverage, token-level similarity, sequence-based similarity, and overlap of character trigrams. These values are combined in a weighted sum. The highest weights are given to soft query coverage and token-level overlap, because these measures indicate most directly whether the query is covered by the document text.

After the global scoring, a local verification is carried out for the eight best ranked documents. The query is compared with the document title and with the single sentences of the abstract. A document is only accepted if the best local text span gives clear support for the match, for example by:

- high query coverage,
- high partial sequence similarity, or
- near-exact containment.

At the end, an ambiguity check is applied, so that only sufficiently clear cases are returned:

- A prediction is only returned if exactly one document passes the local verification.
- The accepted document has to be separated from the next-best candidate by a score margin of at least 0.025.
- If several documents are still similarly plausible, or if no document has enough local support, the detector abstains.

8.3. Token Similarity Detector Details

The token similarity detector uses Unicode *NFKC* normalization and lowercasing before the text is tokenized. For the tokenization, the regular expression `[a-z0-9]+` is used. Afterwards, stopwords are removed. Numeric tokens are removed as well, except if they are four-digit numbers. Non-numeric tokens with fewer than three characters are also excluded. The remaining tokens are treated as signal tokens and are weighted with IDF:

$$\mathrm{idf}(t) = \log \frac{1 + N}{1 + \mathrm{df}(t)} + 1,$$

where N is the number of documents and $\mathrm{df}(t)$ is the document frequency of token t.

For each query, the signal tokens are ordered by decreasing IDF. The rarest tokens are used as the signature of the query, while the following tokens are kept as reserve evidence. Two settings are used:

- In the balanced setting, the query has to contain at least six distinct signal tokens. The signature consists of four tokens, each with an IDF of at least 1.60, and three additional tokens are kept as reserve tokens.
- In the strict setting, the query has to contain at least seven distinct signal tokens. The signature consists of five tokens, each with an IDF of at least 1.90, and four additional tokens are kept as reserve tokens.

A document is only considered as a candidate if it satisfies all signature-based checks. These checks make sure that the document contains the main rare tokens of the query and that they also occur close enough in the document text:

- All signature tokens have to be present in the document.
- The document has to cover enough of the combined IDF mass of the signature and reserve tokens.
- There has to be a local span in the document which contains all signature tokens.

The required IDF-mass coverage depends on the setting:

- In the balanced setting, the total-mass threshold is 0.78 and the local-mass threshold is 0.72.
- In the strict setting, the total-mass threshold is 0.86 and the local-mass threshold is 0.80.

The detector abstains if no document passes these checks. It also abstains if too many documents remain possible:

- In the balanced setting, at most three documents may pass the checks.
- In the strict setting, at most two documents may pass the checks.

Documents which are accepted after these filters are ranked by local coverage, total coverage, reserve-token hits, and document ID. If semantic reranking is enabled, these accepted documents are then ordered again with the semantic retriever.

8.4. Author Detector Details

The author detector uses language-specific lexical lists for English, German, and French.

For English, author cues include *by*, *author*, *authors*, *according to*, *co authored*, *coauthored*, *prof*, *professor*, and *dr*. Author-reporting verbs include *argues*, *claims*, *concludes*, *describes*, *examines*, *finds*, *notes*, *reports*, *shows*, *states*, *suggests*, and *writes*. Supportive author-work terms include *study*, *paper*, *analysis*, *review*, *preprint*, *article*, *publication*, *findings*, *report*, *work*, *research*, *dataset*, and *model*.

For German, author cues include *von*, *laut*, *nach*, *autor*, *autorin*, *autoren*, *verfasst von*, and *geschrieben von*. Author-reporting verbs include *argumentiert*, *behauptet*, *beschreibt*, *erklaert*, *findet*, *schreibt*, *sagt*, and *zeigt*. Blocker terms include *gesundheit*, *hochschule*, *institut*, *klinik*, *medizin*, *universitat*, *virologie*, and *wissenschaft*.

For French, author cues include *de*, *selon*, *d'apres*, *auteur*, *auteure*, *auteurs*, *redige par*, and *ecrit par*. Author-reporting verbs include *affirme*, *argumente*, *decrit*, *ecrit*, *estime*, *explique*, *indique*, *montre*, *souligne*, and *trouve*. Blocker terms include *clinique*, *hopital*, *institut*, *medecine*, *recherche*, *sante*, *universite*, and *virologie*.

Shared blocker terms include institution or affiliation followers such as *university*, *college*, *hospital*, *institute*, *center*, *centre*, *lab*, *laboratory*, and *department*; collective-author terms such as *group*, *team*, *network*, *committee*, *consortium*, and *collaboration*; and common collision terms such as *trump*, *fauci*, *lancet*, *jama*, *cox*, *cardiology*, *charite*, *corona*, *epidemiology*, *immunology*, *microbiology*, *moderna*, *neurology*, *psychiatry*, *therapeutics*, and *virology*.

8.5. Title Matcher Details

The title matcher first normalizes the title and query text with Unicode *NFKC* normalization and lowercasing. Afterwards, the text is tokenized with the pattern `[a-z0-9]+`. The token sequences of all titles are stored in a trie. With this structure, exact title matches can be found against contiguous spans in the query.

Signal tokens are the title tokens which are not stopwords. The stopword list contains *a*, *an*, *and*, *are*, *as*, *at*, *be*, *by*, *for*, *from*, *in*, *into*, *is*, *it*, *of*, *on*, *or*, *that*, *the*, *to*, *was*, *were*, and *with*. Title matches with fewer than three signal tokens are ignored.

For every remaining title match, the following score is calculated:

$$\texttt{token\_len} \cdot 100 + \texttt{hit\_count}.$$

For each document id, only the best score is kept.

If there is exactly one candidate document and the best matched title has fewer than six signal tokens, the matcher can extend the result to similar titles. This expansion is only done for titles with at least eight shared filtered tokens and an overlap ratio of at least `0.70`. Expanded candidates are scored with:

$$50000 + \texttt{shared\_tokens} \cdot 100 + \texttt{token\_len}.$$

The final candidate list is limited before a result is returned:

- At most five final candidates are kept.
- If no candidate remains, the matcher abstains.
- If more than five candidates remain, the matcher also abstains.

The matcher also uses a blocklist for false positives from short titles. A title match is blocked if its normalized title key appears in this list and the title has at most six tokens.

For the handling of near-duplicate titles, a graph over all titles in the collection is built. Two titles are connected if they are very similar according to the following conditions:

- They share at least eight signal tokens.
- Their token lengths differ by at most four.
- Their token-set Jaccard similarity is at least `0.88`.

The connected titles are merged with union-find. If a matched title belongs to a cluster with several document ids, all document ids from this cluster are added to the candidate set before scoring and ranking.

If this option is enabled, the final list of candidates can additionally be reranked with a semantic candidate reranker. This reranker uses the dot product between query and document embeddings.

8.6. Journal Detector Details

The journal detector uses language-specific lexical lists for English, German, and French.

For English, publication-venue cues include *published in*, *released in*, *appeared in*, *reported in*, *study in*, *paper in*, *article in*, *preprint in*, and *review in*. Strong journal patterns include *journal X*, *in the journal X*, and *in journal X*. Citation markers include *pubmed*, *pmid*, *doi*, *nih*, *pmc*, *medrxiv*, *biorxiv*, *volume*, *issue*, and *vol*.

For German, publication-venue cues include *veroeffentlicht in*, *erschienen in*, *publiziert in*, *berichtet in*, *studie in*, *papier in*, *artikel in*, *vorabdruck in*, and *uebersicht in*. Citation markers include *band*, *heft*, *ausgabe*, *nummer*, *nr*, and *jahrgang*. German normalization also folds characters such as *ä* to *ae*, *ö* to *oe*, *ü* to *ue*, and *ß* to *ss*.

For French, publication-venue cues include *publie dans*, *publiee dans*, *paru dans*, *parue dans*, *rapporte dans*, *etude dans*, *article dans*, *prepublication dans*, and *revue dans*. Citation markers include *tome*, *numero*, *no*, and *fascicule*. The French variant also applies accent folding during normalization.

Common ambiguous or blocked journal terms include *health*, *medicine*, *science*, *nature*, *blood*, *brain*, *cell*, *gut*, *nutrition*, *radiology*, *virology*, and *immunology*. Further low-precision discipline terms include *clinical*, *infectious*, *epidemiology*, *neurology*, *oncology*, *pediatrics*, *psychiatry*, and *radiology*. Examples of generic two-word journal names that require stronger evidence are *public health*, *clinical medicine*, *internal medicine*, *physical therapy*, *preventive medicine*, and *psychological medicine*.

Language-specific blocker terms include German terms such as *gesundheit*, *medizin*, *klinik*, *wissenschaft*, *forschung*, *virologie*, and *oeffentliche gesundheit*, as well as French terms such as *sante*, *medecine*, *clinique*, *recherche*, *virologie*, and *sante publique*.

The detector also uses journal-family mappings. Mentions of broad families such as *lancet*, *bmj*, *jama*, *nature*, *science*, and *health affairs* can be expanded to related journal titles when the context supports this interpretation.

8.7. Quote Detector Details

The quote detector is mainly based on quotation-mark patterns and on a minimum length for quoted text. The basic version recognizes straight double quotes, straight single quotes, and curly double quotes. This includes forms such as *"..."*, *'...'*, and *“...”*. For German, additional quotation forms are supported, for example *„...“*, *‚...‘*, *»...«*, and *«...»*. For French, guillemets are also supported, especially *«...»*.

A quoted span is only used when it contains at least five normalized words. Hyphenated compounds, such as *state-of-the-art*, are counted as one word. The actual matching is done by exact normalized substring search. For this, the quoted text is searched in document titles and in the full document texts. If a references section can be detected, the full text is cut before this section, so that matches inside references are not used.

If the exact quote matching gives more than three different candidate documents, the detector abstains. Otherwise, the candidates are ordered by:

- the number of matched quotes, and
- the document id.

The detector can also extend the candidate set by using title siblings. Documents with the same normalized title are added directly. Similar titles can also be added, but only if they fulfil the following conditions:

- They share at least six signal tokens.
- Their shared-token ratio is at least `0.70`.

For this title-based expansion, signal tokens from titles have to be at least three characters long and must not be stopwords. Four-digit years are allowed as an exception.

8.8. Named Entity Detector Details

The named entity detector links entities to Wikidata QIDs. For this, *spaCy*, *en_core_web_sm*, and *spacy-entity-linker* are used [26, 27, 28]. The same detector is used for the English, German, and French pipelines. For non-English queries, the detector depends on the upstream translation and does not use separate language-specific entity-linking rules.

The detector works with QID overlap instead of lexical cue words. A query is only used if enough linked entities are found:

- The query must contain at least 2 linked entities.
- A candidate document must share at least 2 entities with the query.
- At least one shared entity must be rare.

An entity is counted as rare if it appears in at most 10% of the indexed documents. The following thresholds are used:

- Minimum linked query entities: 2.
- Minimum shared entities: 2.
- Maximum matching documents: 5.
- Maximum candidates: 3.
- Maximum rare-document fraction: 0.10.
- Minimum shared entities for the top candidate: 3.

After filtering, the remaining candidates are ranked in a simple order:

- first by the number of shared entities,
- then by the document id as a tie-breaker.

The top candidate must share at least 3 entities with the query. If another candidate has the same number of shared entities as the top candidate, the detector abstains when unique-top-candidate filtering is enabled.

8.9. Fine-Tuning Details for Semantic Retrieval

- **Epochs:** `1`
- **Learning rate:** `1.5e-5`
- **Weight decay:** `0.01`
- **Warmup ratio:** `0.05`
- **Per-device batch size:** `2`
- **Gradient accumulation steps:** `8`
- **Effective batch size:** `16`
- **Global steps:** `1873`
- **Training examples:** `29954`
- **Query max length:** `96`
- **Document max length:** `320`