ATOM@CheckThat!: Retrieve the Implicit - Scientific Evidence Retrieval

Notebook for the CheckThat Lab at CLEF 2025

Moritz Staudinger[1,*], Alaa El-Ebshihy[1,2], Wojciech Kusa[3], Florina Piroi[1,2] and Allan Hanbury[1]

[1]*Technische Universität Wien, Austria*
[2]*Research Studios Austria, Data Science Studio, Austria*
[3]*NASK National Research Institute, Poland*

Abstract

With the rapid growth of scientific publications and their increasing circulation through social media, tracing claims back to their original sources has become increasingly difficult. The CheckThat! 2025 Claim Source Retrieval task addresses this challenge by requiring systems to retrieve the scientific paper implicitly referenced in a social media post. In this paper, we present our approach to solving this task. We explore a two-stage retrieval pipeline that combines lexical and dense retrievers with both pointwise and listwise re-rankers. We evaluate the impact of enriching document representations with full-text and summary content from CORD-19, and analyze trade-offs between model size, retrieval effectiveness, and runtime.

Keywords

Fact Checking, Information Retrieval, Scientific Document Retrieval, Listwise Ranking

1. Introduction

In our data-driven world, the volume of published literature – including newspaper articles, social media posts, and scientific publications – is increasing at a rapid pace [1]. With the exponential growth in scientific articles each year, it becomes increasingly difficult to stay informed about the latest developments and discoveries across different fields [2]. For example, over a thousand machine learning papers are submitted to arXiv each month[1]. This challenge affects not only researchers but also the general public, especially as scientific findings are frequently discussed on social media platforms like X (formerly Twitter)[2]. In such cases, posts often refer to scientific work without proper citations or links, making it difficult to identify the original source and assess its reliability.

Identifying the exact paper being referenced is a crucial first step in finding appropriate evidence for scientific claims being discussed and in detecting potential misinformation. Therefore, choosing an effective retrieval method for this task lays the foundation for automating and improving the scientific fact-checking process. Fact-checking is the process of assessing the validity of a claim by verifying it against credible sources of information. In the context of scientific text, this often involves identifying relevant evidence from academic literature to support or refute a given claim. Specifically, scientific fact-checking focuses on verifying claims grounded in scientific knowledge, typically using corpora of scientific publications.

The first step in the scientific fact-checking pipeline is document retrieval – identifying relevant scientific publications from a corpus that may contain evidence related to the claim. Subtask 4b of the CheckThat! 2025 lab, *Scientific Claim Source Retrieval*, focuses specifically on this retrieval step. The

CLEF 2025 Working Notes, September 9 – 12 September 2025, Madrid, Spain
*Corresponding author.
moritz.staudinger@tuwien.ac.at (M. Staudinger); alaa.el-ebshihy@tuwien.ac.at (A. El-Ebshihy); wojciech.kusa@nask.pl (W. Kusa); florina.piroi@tuwien.ac.at (F. Piroi); allan.hanbury@tuwien.ac.at (A. Hanbury)
0000-0002-5164-2690 (M. Staudinger); 0000-0001-6644-2360 (A. El-Ebshihy); 0000-0003-4420-4147 (W. Kusa); 0000-0001-7584-6439 (F. Piroi); 0000-0002-7149-5843 (A. Hanbury)

[1]https://arxiv.org/stats/monthly_submissions
[2]https://x.com/

Scientific Claim Source Retrieval task is defined as follows: given an implicit reference to a scientific paper, specifically a social media post on Twitter that mentions a research publication without a URL, the goal is to retrieve the mentioned paper from a pool of candidate scientific documents.

In this work, we present out approaches to solve the CheckThat! *Scientific Claim Source Retrieval* challenge at CLEF 2025 [3]. We evaluated the impact of fine-tuned pointwise ranker models, in contrast to open-domain listwise ranker models. We further investigated if the full-text or the summary of the scientific evidence is beneficial for the ranking stage.

The rest of this paper is structured as follows: in Section 2 we discuss previous approaches to scientific claim retrieval. In Section 3 we discuss our approaches to retrieve the most relevant documents, while Section 4 presents our results for the CheckThat! Lab. In Section 5 we discuss our findings, before we conclude our work in Section 6.

2. Related work

Scientific fact-checking task is a variation of the general fact-checking process, focusing specifically on assessing claims of scientific knowledge [2]. While the primary role of general fact-checking is to detect misinformation, scientific fact-checking aims to verify research hypotheses and identify supporting evidence within scientific work. It leverages evidence sources, often derived from scientific publications, to validate claims.

Generally, the framework of scientific fact-checking consists of three main components [2], namely: (1) document retrieval, (2) evidence (rationale) selection, and (3) verdict prediction [4]. *Document retrieval* involves identifying relevant documents from a scientific corpus that may contain evidence related to the claim, using sparse or dense retrieval models. *Evidence selection*, then, extracts specific sentences or passages from these documents that can serve as rationales to support or refute the claim. Finally, *verdict prediction* uses the selected evidence to classify the claim's veracity, typically into categories such as *Supported*, *Refuted*, or *Not Enough Information*. CheckThat! Task 4b focuses on the document retrieval component, where the aim is to retrieve the specific scientific publication implicitly referenced in a tweet, from a pool of candidate papers. In the following, we review previous work on document retrieval approaches relevant to this task, including both traditional and neural methods.

Vladika et. al. [2] presents a comprehensive overview of the scientific fact-checking task, covering its main components, available datasets, and different approaches employed to address each stage of the pipeline. In the context of the document retrieval, the goal is to identify relevant documents from a scientific corpus that may contain evidence related to a given claim. This components is typically solved with Information Retrieval (IR) techniques, which are categorized into sparse and dense retrieval approaches. Sparse retrieval approaches, such as TF-IDF and BM25, rely on keyword matching through inverted indexes. On the other hand, dense retrieval approaches use dense vector representations of queries and documents, enabling retrieval based on semantic similarity [5].

One of the most commonly used datasets in scientific fact-checking, and the basis for many developed models, is the SCIFACT dataset [6]. The document retrieval task involves retrieving relevant abstracts from a corpus of around 5k scientific abstracts from the biomedical domain. The baseline model, VeriSci, relied on simple TF-IDF scoring to retrieve the top-k abstracts. More advanced models, such as VerT5erini and MultiVerS, adopted a two-stage retrieval strategy: first using BM25 to retrieve candidate abstracts, followed by neural re-ranking with a T5-based model trained on the MS MARCO passage dataset [7, 8]. Other approaches, including ParagraphJoint and ARSJoint, utilized dense retrieval with BioSentVec embeddings [9], which were trained on a large corpus of biomedical texts. The SCIFACT corpus was later expanded to 500k documents [10], resulting in a noticeable drop in overall fact-checking performance. This highlights the limitations of existing retrieval methods and the need for more effective semantic search techniques. The COVID-Fact dataset[11] addresses this challenge by retrieving snippets from the top 10 results returned by the Google Search API for a given claim, mimicking a more realistic evidence-gathering process.

To the best of our knowledge, MultiVerS achieves state-of-the-art performance on the SCIFACT

dataset. Inspired by this approach, we use a BM25 followed by T5-based re-ranking as a baseline in our experiments.

3. Method

As previously discussed, retrieval approaches for fact-checking typically achieve the best performance when following a two-stage strategy. This consists of an initial fast retrieval component that retrieves candidate documents for a given claim, followed by a reranking component that produces a final ranked list of documents.

In our work, we employed several retrieval and ranking strategies and also varied the type of information provided to our pipeline. The following subsections describe the individual components of our system.

3.1. Data

3.1.1. Preprocessing

To improve the retrieval performance of our lexical models, we applied a series of preprocessing steps to both the input tweets and the scientific documents. Specifically, we removed URLs and mentions from the tweets, and further cleaned the text by removing special characters (e.g., hashtags and punctuation), digits, and non-ASCII characters. Afterward, we tokenized the text, removed stopwords, and applied stemming to all tokens.

We followed a similar preprocessing strategy for our embedding-based models, particularly for removing URLs and special characters, as this was found to improve retrieval performance.

3.1.2. Data Enrichment

The subset of the CORD-19 dataset provided by the lab organizers includes only the abstracts of scientific papers. However, the full CORD-19 dataset also contains full-text information for many publications. Since inspecting the full-text can be valuable for scientific fact-checking, we analyzed the impact of incorporating additional information—either in the form of summaries or full-texts into both the retrieval and ranking stages.

To this end, we utilized the full-text CORD-19 dataset (approximately 19GB in size) and linked its full-text entries to the retrieval subset used in CheckThat! Task 4B using the `cord_uid` field.

During this process, we found that only about 127,000 documents in the full CORD-19 dataset contain full-text information, out of a total of roughly 369,000 documents. The coverage was similar, slightly higher, in the provided subset as 2,759 out of 7,764 entries (approximately 35%) included a full-text version.

Despite this limitation, we aimed to test our hypothesis that incorporating either full-text or paper summary, instead of the paper abstract, could improve retrieval performance. We used dense representations of both abstracts and full-texts to extract the most semantically similar paragraphs and generated summaries that were, on average, twice the length of the original abstracts. For generating these summaries, we used the *SentenceTransformer* library with the *all-MiniLM-L6-v2*[3] model.

3.2. Retrieval

For the retrieval stage, we implemented one lexical model in BM25 (the rank_bm25[4] implementation) and two dense bi-encoder models:

- `gtr-t5-base`[5]

[3]https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
[4]https://pypi.org/project/rank-bm25/
[5]https://huggingface.co/sentence-transformers/gtr-t5-base

- `gtr-t5-large`[6]

We hypothesized that both dense models will outperform BM25, as fine-tuned dense retrieval models generally achieve superior performance compared to lexical methods [12, 13]. In particular, we are interested in analyzing how model size affects retrieval effectiveness in the context of fact-checking, and whether improvements at the retrieval stage would also translate into gains during reranking.

To test this, we fine-tuned both dense models on the provided CORD-19 subset using the *Sentence-Transformer* library. We employed the *MultipleNegativesRankingLoss* loss function, training over five epochs with a learning rate of $2 \cdot 10^{-5}$ on all available training queries.

In addition, we implemented an ensemble retrieval approach by combining `gtr-t5-large` with BM25 using reciprocal rank fusion. Such hybrid strategies—combining lexical and dense retrieval have been shown to improve performance, as demonstrated by Althammer et al. [14] in the context of Dense Passage Retrieval for legal texts.

3.3. Ranking

For the reranking stage, we investigated the effectiveness of a smaller, domain-specific pointwise ranker model—`monot5-base-med-msmarco` in comparison to larger and more resource-intensive models.

With the emergence of large language models (LLMs) since 2022 and the expansion of context windows, listwise ranking strategies have gained popularity. These approaches typically outperform pointwise and pairwise methods, as they consider a ranked list of documents at once, rather than evaluating documents in isolation or in pairs. As pointwise rankers compute scores for each query-document pair independently, and therefore the computation time increases linearly. In contrast to this, listwise models (e.g., MXBAI) process an entire document list at once, yielding relatively stable runtimes across different list sizes.

Therefore, we included three additional reranking models in our experiments:

- `mxbai-rerank-base-v2` (listwise)
- `mxbai-rerank-large-v2` (listwise)
- `monot5-3b-inpars-v2-trec_covid` (pointwise, trained on CORD-19)

Given that the CheckThat! Task 4 dataset is derived from the medical domain (CORD-19), we prioritized medical-domain language models for the pointwise rerankers.

We fine-tuned both MonoT5 models—training the larger model for one epoch and the smaller one for three epochs—as we observed no significant performance improvement with further training. For supervision, we constructed training samples consisting of one relevant document and two hard negatives sampled from the top-5 BM25 retrieval results.

4. Experiments and Results

4.1. Dataset

The CheckThat! Lab [3] provided a subset of the CORD-19 collection as a retrieval collection, along with a set of tweets and their corresponding relevance labels as queries.

The document collection contains 7,764 distinct entries and 17 columns. These features include fields such as *title* and *abstract*, as well as metadata fields. The metadata fields include: *pmc_id*, *pubmed_id*, *license*, *publish time*, *authors*, *journal*, *mag_id*, *arxiv_id*, *label*, *time* and a *timet* id.

We excluded these metadata fields, as they were less relevant for retrieval performance and did not add additional information. We enriched this corpus by linking to the CORD-19 full-text dataset and generating summaries based on this additional information (see Section 3.1.2 for details).

[6] https://huggingface.co/sentence-transformers/gtr-t5-large

The tweet dataset is divided into three splits: *train*, *dev*, and *test*. Each tweet is associated with exactly one relevant scientific document. The *train* set contains 12,853 tweets, while the *dev* and *test* sets contain 1,400 and 1,446 tweets, respectively.

4.2. Experimental Evaluation

We evaluated our retrieval and reranking pipeline on the development and test sets of the shared task by reranking the top 10, 20, and 50 retrieved candidates. Table 1 presents the results on the development set. The initial retrieval stage achieved an *MRR@10* of 0.6130 for BM25 and up to 0.6475 for *GTR-T5-Large*, with only marginal gains beyond the top 10 positions.

We observed a similar trend when applying reranking: most relevant documents were found within the top 5 positions, and performance improved only slightly when extending the reranking depth. For example, in the *GTR-T5 + MonoT5 + Abstract* setting, MRR increased by only 0.5% when reranking 10 documents and by approximately 1% at a depth of 50. This pattern aligns with the behavior of the MRR metric, which is more sensitive to high-ranking positions.

Table 1
MRR performance on the development set, showing retriever, input type, reranker, and reranking size.

Retriever	Information	Ranker	Size	MRR@1	MRR@5	MRR@10	MRR@50
BM25	Abstract	-	-	0.5514	0.6076	0.6130	0.6182
Ensemble	Abstract	-	-	0.5336	0.6262	0.6341	0.6387
GTR-T5-Base	Abstract	-	-	0.5600	0.6226	0.6302	0.6359
GTR-T5-Large	Abstract	-	-	0.5714	0.6404	0.6475	0.6527
GTR-T5-Large	Abstract	MonoT5-med-base	10	0.6579	0.7014	0.7061	0.7061
			20	0.6607	0.7063	0.7109	0.7136
			50	0.6571	0.7020	0.7065	0.7109
GTR-T5-Large	Full-text	MonoT5-med-base	10	0.6607	0.7026	0.7075	0.7075
			20	0.6543	0.7014	0.7065	0.7093
			50	0.6507	0.696	0.7003	0.7049
GTR-T5-Large	Summary	MonoT5-med-base	10	0.6600	0.7028	0.7074	0.7074
			20	0.6514	0.7001	0.7065	0.7093
			50	0.6443	0.6918	0.7003	0.7050
GTR-T5-Large	Abstract	MonoT5-med-3B	10	0.6714	0.7124	0.7124	0.7124
			20	0.6764	0.7236	0.7279	0.7296
			50	0.6836	0.7317	0.7357	0.7393
GTR-T5-Large	Abstract	MXBAI-base-v2	10	0.7100	0.7580	0.7580	0.7580
			20	0.7100	0.7580	0.7613	0.7637
			50	0.7100	0.7580	0.7613	0.7648
GTR-T5-Large	Abstract	MXBAI-large-v2	10	0.7243	0.7721	0.7758	0.7758
			20	0.7243	0.7721	0.7770	0.7785
			50	0.7243	0.7721	0.7770	0.7793

We also analyzed the runtime efficiency of different retriever–ranker combinations based on the dev split of the provided dataset, as shown in Table 2. For the runtime evaluation we run our systems on a server with 4 cores of a AMD EPYC 7413 CPU, an A40 GPU and 64 GBs of RAM.

First-stage retrieval models were efficient, requiring between 0.009 and 0.045 seconds per query. In contrast, rerankers varied substantially in speed, depending on model size and reranking depth. This is expected: the *MonoT5-med-base* model has 220 million parameters, while *MonoT5-med-3B* has 3 billion. The listwise rerankers (MXBAI) yielded stable runtime across different list sizes, as discussed before.

Although runtime is not the primary concern in fact-checking—where accuracy is paramount—it is still valuable to understand the trade-off between computational cost and retrieval quality.

Table 2
Retrieval time for different retriever–ranker combinations

Retriever	Ranker	Size	Time (s)	Time/query (s)
BM25 (CPU)	-	50	37.97	0.027
Ensemble	-	50	62.95	0.045
GTR-T5-Base	-	50	12.33	0.009
GTR-T5-Large	-	50	23.20	0.017
GTR-T5-Large	MonoT5-med-base	10	346.72	0.248
		20	669.58	0.478
		50	1629.59	1.164
GTR-T5-Large	MonoT5-med-3B	10	1583.50	1.131
		20	3145.29	2.247
		50	7861.30	5.615
GTR-T5-Large	MXBAI-base-v2	10	2637.34	1.884
		20	2637.71	1.884
		50	2649.95	1.893
GTR-T5-Large	MXBAI-large-v2	10	5555.63	3.968
		20	5556.99	3.969
		50	5563.73	3.974

While the combination of *GTR-T5-Large* with *MXBAI-large-v2* achieved the best performance overall (MRR@5 = 0.7721), it required nearly twice the computation time compared to the smaller *MXBAI-base-v2*, which still performed competitively. For this reason, we selected the smaller model for our evaluation runs.

We submitted eight runs for the leaderboard, summarized in Table 3. Our baseline run, combining BM25 with the smaller MonoT5 model, achieved an MRR@5 of 0.6262. Subsequent runs aimed to quantify how improvements in retrieval quality affected final ranking performance.

Although dense retrievers outperformed BM25 by 0.02 to 0.035 on the development set, these gains diminished after reranking, where differences across configurations fell below 0.01—an effect also visible in the evaluation of Schlatt et al. [15].

Using full-text and summary inputs during ranking led to decreased MRR compared to using abstracts alone. However, performance improved when using the larger MonoT5 model, with a gain of approximately 0.015 on the test set when reranking 50 documents. This matches our observations on the development set, where the larger model consistently outperformed the smaller one by 0.01 to 0.03. Our best-performing configuration used the listwise reranker *MXBAI-base-v2*, which achieved an MRR@5 of 0.6568.

Table 3
Results on the test set for all submitted runs

Retriever	Information	Ranker	Size	MRR@5
BM25	Abstract	MonoT5-med-base	50	0.6262
GTR-T5-Base	Abstract	MonoT5-med-base	50	0.6286
GTR-T5-Large	Abstract	MonoT5-med-base	50	0.6336
GTR-T5-Large	Summary	MonoT5-med-base	50	0.6195
GTR-T5-Large	Full-text	MonoT5-med-base	50	0.6195
GTR-T5-Large	Abstract	MonoT5-med-3B	20	0.6310
GTR-T5-Large	Abstract	MonoT5-med-3B	50	0.6415
GTR-T5-Large	Abstract	MXBAI-base-v2	20	**0.6568**

5. Discussion

In this section, we present a discussion and findings from the results of our experiments.

Effect of size of reranked candidates We observed that the performance slightly changed or constant while increasing the reranking depth. This shows that most relevant documents are found withing the top 5 to 10 candidates. This suggests that limiting reranking to a smaller candidate set may be sufficient for effective performance, enabling more efficient use of computational resources.

Model size vs. time We also found that model size affects the retrieval effectiveness, in terms of performance and time. Larger rerankers, such as MonoT5-med-3B and MXBAI-large-v2, consistently outperformed smaller models. However, these gains come at the cost of increased runtime and memory usage.

Pointwise vs. Listwise rerankers Our comparison shows that, although the listwise reranking approaches outperforms the pointwise alternatives, it comes at the cost of the retrieval time. However, they are especially suitable for reranking small candidate sets given there increased performance on sets of size 10.

Full-text and Summary vs. Abstract retrieval Our results indicate that enriching the retrieval corpus with full-text content or extracted summaries does not necessarily improve performance over using abstracts alone. In some cases, these additional inputs even introduced noise and reduced ranking quality. This suggests that naive enrichment strategies may not yield benefits and that future work should explore more targeted methods, such as using section-aware embeddings or fine-tuned summarization models.

Relevant Documents After failing to improve the ranking performance beyond an MRR@5 of 78

To this end, we plotted the ranks of our retrieved documents across all queries. Figure 1 shows that most relevant documents are already ranked highly: over 800 are ranked first, and 1300 out of 1400 relevant documents appear within the top 50 results—the subset used for reranking. However, this also implies that our reranker cannot affect the remaining 100 relevant documents, as they fall outside the reranking range (i.e., below rank 50). Notably, some relevant documents are ranked as low as position 300.

Figure 2 demonstrates that our reranker improved the rankings for many relevant documents: 1014 are now ranked first, and 87 are ranked second. Still, 100 relevant documents are absent from the top 50 entirely.

This analysis highlights an opportunity to focus on improving the retrieval of long-tail research documents, which could further enhance overall ranking performance.

6. Summary and Future Work

In this work, we presented our approach and results for the CheckThat! Claim Retrieval Shared Task. We evaluated the impact of three different first-stage retrieval algorithms and four different ranking models. Additionally, we examined whether augmenting the input with full-text or summary information could improve performance. Contrary to our expectations, we found that this additional information negatively affected retrieval effectiveness, but this could be due to missing data and needs further exploration.

Furthermore, our results show that open-domain listwise rerankers consistently outperform domain-specific pointwise rerankers even those with significantly larger model sizes. This suggests that ranking strategy plays a more crucial role than model size or domain specialization in this task.

Figure 1: Distribution of relevant documents retrieved by our GTR-T5-Large first-stage retriever

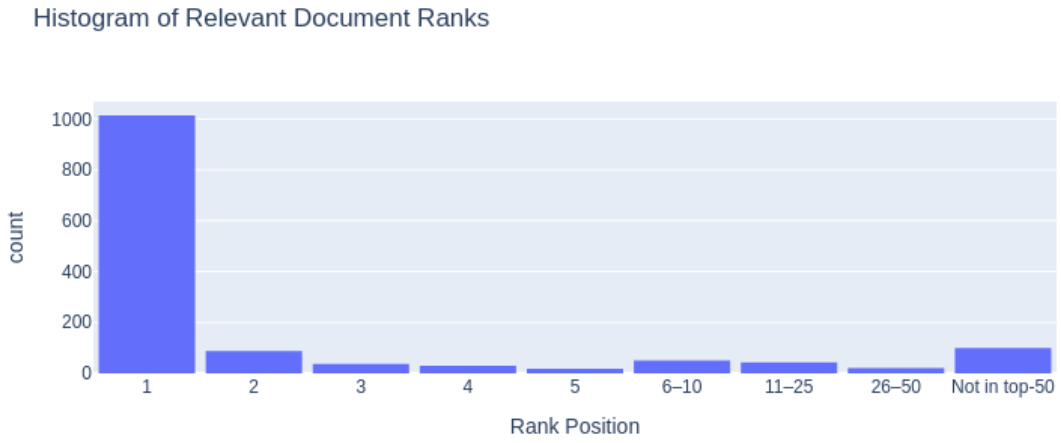

Figure 2: the distribution of the relevant documents for our best performing Retrieval+Reranker combination (GTR-T5-Large, MXBAI-large-v2)

For future work, we propose exploring the use of an additional retrieval or classification layer applied to the top five retrieved documents. This could help refine the final ranking and improve overall performance, by improving the ranking of the around 200 documents which are in the top 5 but not ranked top . Such an approach would allow the use of larger models and more detailed input representations in a computationally feasible way.

A more important objective would be to improve the retrieval of the long-tail of research documents, which are retrieved and ranked by our first-stage retrieval engine between rank 50 and rank 300. Therefore, it is important to investigate why these documents are not ranked higher in the first-stage.

Another promising direction would be to construct or extend datasets that include full-text versions of the cited scientific papers. This would enable a more systematic evaluation of how different information granularities abstracts, summaries, and full-texts impact both retrieval and ranking performance.

Declaration on Generative AI

During the preparation of this work, the author(s) used GPT-4o in order to: check grammar and spelling. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] M. Staudinger, A. El-Ebshihy, A. M. Ningtyas, F. Piroi, A. Hanbury, AMATU@ SimpleText2024: are LLMs any good for scientific leaderboard extraction, in: G. Faggioli, N. Ferro, P. Galuščáková, A. G. S. de Herrera (Eds.), Working Notes of CLEF 2024 - Conference and Labs of the Evaluation Forum, CEUR Workshop Proceedings, CEUR-WS, Online, 2024.

[2] J. Vladika, F. Matthes, Scientific fact-checking: A survey of resources and approaches, in: A. Rogers, J. Boyd-Graber, N. Okazaki (Eds.), Findings of the Association for Computational Linguistics: ACL 2023, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 6215–6230. URL: https://aclanthology.org/2023.findings-acl.387/. doi:10.18653/v1/2023.findings-acl.387.

[3] 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, ????

[4] X. Zeng, A. S. Abumansour, A. Zubiaga, Automated fact-checking: A survey, Language and Linguistics Compass 15 (2021) e12438.

[5] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 6769–6781. URL: https://aclanthology.org/2020.emnlp-main.550/. doi:10.18653/v1/2020.emnlp-main.550.

[6] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, H. Hajishirzi, Fact or fiction: Verifying scientific claims, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 7534–7550. URL: https://aclanthology.org/2020.emnlp-main.609/. doi:10.18653/v1/2020.emnlp-main.609.

[7] R. Nogueira, Z. Jiang, R. Pradeep, J. Lin, Document ranking with a pretrained sequence-to-sequence model, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 708–718. URL: https://aclanthology.org/2020.findings-emnlp.63/. doi:10.18653/v1/2020.findings-emnlp.63.

[8] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, L. Deng, Ms marco: A human generated machine reading comprehension dataset (2016). URL: https://www.microsoft.com/en-us/research/publication/ms-marco-human-generated-machine-reading-comprehension-dataset/.

[9] Q. Chen, Y. Peng, Z. Lu, BioSentVec: creating sentence embeddings for biomedical texts , in: 2019 IEEE International Conference on Healthcare Informatics (ICHI), IEEE Computer Society, Los Alamitos, CA, USA, 2019, pp. 1–5. URL: https://doi.ieeecomputersociety.org/10.1109/ICHI.2019.8904728. doi:10.1109/ICHI.2019.8904728.

[10] D. Wadden, K. Lo, B. Kuehl, A. Cohan, I. Beltagy, L. L. Wang, H. Hajishirzi, SciFact-open: Towards open-domain scientific claim verification, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2022, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 4719–4734. URL: https://aclanthology.org/2022.findings-emnlp.347/. doi:10.18653/v1/2022.findings-emnlp.347.

[11] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Eide, K. Funk, R. Kinney, Z. Liu, W. Merrill, P. Mooney, D. A. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A. D. Wade, K. Wang, C. Wilhelm, B. Xie, D. Raymond, D. S. Weld, O. Etzioni, S. Kohlmeier, CORD-19: the covid-19 open research dataset, CoRR abs/2004.10706 (2020). URL: https://arxiv.org/abs/2004.10706. arXiv:2004.10706.

[12] J. Ni, C. Qu, J. Lu, Z. Dai, G. Hernandez Abrego, J. Ma, V. Zhao, Y. Luan, K. Hall, M.-W. Chang,

Y. Yang, Large dual encoders are generalizable retrievers, in: Y. Goldberg, Z. Kozareva, Y. Zhang (Eds.), Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 2022, pp. 9844–9855. URL: https://aclanthology.org/2022.emnlp-main.669/. doi:10.18653/v1/2022.emnlp-main.669.

[13] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 39–48. URL: https://doi.org/10.1145/3397271.3401075. doi:10.1145/3397271.3401075.

[14] S. Althammer, A. Askari, S. Verberne, A. Hanbury, Dossier@coliee 2021: Leveraging dense retrieval and summarization-based re-ranking for case law retrieval, CoRR abs/2108.03937 (2021). URL: https://arxiv.org/abs/2108.03937. arXiv:2108.03937.

[15] F. Schlatt, M. Fröbe, H. Scells, S. Zhuang, B. Koopman, G. Zuccon, B. Stein, M. Potthast, M. Hagen, Set-encoder: Permutation-invariant inter-passage attention for listwise passage re-ranking with cross-encoders, in: C. Hauff, C. Macdonald, D. Jannach, G. Kazai, F. M. Nardini, F. Pinelli, F. Silvestri, N. Tonellotto (Eds.), Advances in Information Retrieval, Springer Nature Switzerland, Cham, 2025, pp. 1–19.