Advanced Information Retrieval
Exercise 1: SciWeb Claim-Source Retrieval

Dzhamilia Kulikieva (12340251)
Aleksandra Grishan (52209173)
Derek Vlaanderen (12450756)
Agon Sylejmani (01556207)
Alexandru-Bogdan Tarau (12443967)
Group 11

May 2025

Contents

Abstract

We address the task of retrieving scientific papers implicitly referenced in tweets from the CORD-19 collection. We implemented and compared three retrieval approaches, finding that neural methods outperform the traditional baseline.

1 Introduction

The task of Scientific Claim Source Retrieval focuses on retrieving the scientific paper implicitly referenced in a social media post, such as a tweet, from a collection of scientific publications. Unlike explicit references that include direct links or citations, this task requires identifying the most relevant publication based solely on the text of the tweet.

The objective of this project is to implement and evaluate different information retrieval approaches for solving this task. We aim to explore how well traditional and neural retrieval methods can address the challenge of implicit references in short, informal text.

To this end, we implemented and compared three different approaches:

- Traditional IR methods, including a BM25-based retrieval model and a from-scratch TF-IDF pipeline using cosine similarity
- Dense retrieval approach using fine-tuned Sentence-BERT with FAISS indexing
- Neural re-ranking models using a Cross-Encoder (ms-marco-MiniLM-L-6-v2), both in a zero-shot setting and a fine-tuned version trained with pairwise supervision.

Each approach was evaluated in terms of retrieval effectiveness on the development set using the Mean Reciprocal Rank (MRR) metric.

1.1 Local Evaluation Setup

Since we were not registered in time for the CLEF2025 platform, we could not submit our predictions through Codalab. Instead, we used the publicly available file subtask4b_query_tweets_test_gold.tsv, which contains gold annotations for the test set, to locally evaluate each model. This allowed us to compute the MRR scores for the test set using the same metrics defined in the task guidelines.

Additionally, we exported our predictions for the test set in the required TSV format (post_id, preds) to match the expected submission format, as specified in the shared task instructions.

2 Data Description

Dataset	Description
Query set	14,399 tweets with implicit references
	Fields: post_id, tweet_text, cord_uid (train/dev)
Collection set	7,718 CORD-19 papers metadata
	Fields: cord_uid, title, abstract, authors, journal, etc.

Table 1: Overview of the datasets used in Subtask 4b.

In the original dataset, minimal preprocessing was applied. For the collection set, the title and abstract fields were concatenated to form a single text representation for each document. No additional preprocessing was applied to the tweet texts beyond the tokenization required by the retrieval models.

3 Approach 1: Traditional IR Model

3.1 Implementation BM25

We implemented the traditional IR baseline using the BM25Okapi model from the rank_bm25 Python package. The original code was extended with basic preprocessing to improve retrieval effectiveness. The BM25 model was built by concatenating the title and abstract fields of each document in the CORD-19 collection into a single text string. This combined text was used as the indexed field for each document.

The original code used basic whitespace tokenization and did not apply any normalization or filtering. Therefore, we replaced raw .split(' ') with a custom preprocessing function that included lowercasing, punctuation removal, and stopword removal using the built-in English stopword list from scikit-learn

```
import re
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS
stop_words = set(ENGLISH_STOP_WORDS)

def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^a-z0-9\s]', '', text)  # remove punctuation
    tokens = text.split()
    return [word for word in tokens if word not in stop_words]
```

The resulting text was tokenized into words after cleaning. This preprocessing was also applied to each tweet in the query set before computing relevance scores. For each tweet, the BM25 model computed relevance scores between the tokenized query and all documents in the collection. The top 10 ranked documents were selected as candidate articles for evaluation

This baseline model served as a starting point and a reference for comparing the performance of the neural approaches implemented later.

3.2 From-Scratch TF-IDF implementation

To explore a basic traditional IR model from first principles, we implemented a TF-IDF-based retrieval pipeline with cosine similarity, entirely from scratch.

The texts were lowercased, stripped of punctuation, and stopwords were removed using the built-in list from scikit-learn. For each tweet in the query set, TF-IDF vectors were computed, and cosine similarity was used to rank all CORD-19 documents.

```
def cosine_sim(vec1, vec2):
    common = set(vec1) & set(vec2)
    num = sum(vec1[w] * vec2[w] for w in common)
    denom1 = math.sqrt(sum(v**2 for v in vec1.values()))
    denom2 = math.sqrt(sum(v**2 for v in vec2.values()))
    return num / (denom1 * denom2) if denom1 and denom2 else 0
```

The top-10 retrieved articles were then compared against the ground-truth 'cord_uid' annotations using the standard MRR@k evaluation.

3.3 Results

3.3.1 BM25

The BM25 model was evaluated using the MRR metric at different cut-off points (MRR@1, MRR@5, MRR@10). The evaluation was performed on both the training and development sets. The initial results are shown in Table below.

Dataset	MRR@1	MRR@5	MRR@10
Train	0.508	0.551	0.551
Dev	0.505	0.552	0.552

Table 2: Performance of the BM25 baseline model on the train and dev sets.

These results indicate that the BM25 baseline achieves consistent performance on both the training and development sets, with an MRR@5 of approximately 0.55.

To improve upon this baseline, we applied light preprocessing to normalize the input text. This included lowercasing, punctuation removal, and stopword filtering using the built-in stopword list from scikit-learn. After applying these enhancements, the model achieved significantly better performance, as shown in Table 3.

Dataset	MRR@1	MRR@5	MRR@10
Train	0.578	0.627	0.627
Dev	0.579	0.629	0.629

Table 3: Performance of the BM25 baseline model on the train and dev sets.

3.3.2 From-Scratch TF-IDF

The model was evaluated using the same MRR@k metrics as the BM25 baseline. Table 4 shows the performance of the from-scratch TF-IDF model on the train and dev sets.

Dataset	MRR@1	MRR@5	MRR@10
Train	0.470	0.065	0.008
Dev	0.453	0.067	0.008

Table 4: Performance of the from-scratch TF-IDF model on the train and dev sets.

Although the TF-IDF approach reflects the core intuition behind traditional IR methods, its performance is significantly lower than that of the optimized BM25 model (see Table 3). This highlights the practical value of using well-tuned probabilistic models such as BM25 over manually implemented term weighting schemes, especially in real-world scenarios with limited preprocessing and noisy data.

3.3.3 Evaluation on the Test Set

To complete the evaluation, we also applied our improved BM25 model to the test set with annotations. (‘subtask4b_query_tweets_test_gold.tsv‘). The model was not retrained or modified, but applied directly using the same index built from the CORD-19 collection. Table 5 shows the results.

Dataset	MRR@1	MRR@5	MRR@10
Test	0.459	0.519	0.519

Table 5: Performance of the improved BM25 model on the test set with labels.

The BM25 model demonstrated robust generalization performance on the held-out test set, achieving an MRR@5 of 0.519. While slightly lower than the results on the dev set (MRR@5 = 0.629), the drop is expected given the domain shift and absence of any model adaptation to test data. These results confirm the baseline's effectiveness in retrieving relevant scientific documents from short, informal tweet inputs.

We did not evaluate the from-scratch TF-IDF model on the test set. Given its relatively low performance on the training and development sets (see Table 4), we considered it unlikely to yield meaningful results on the test data. Additionally, this allowed us to reduce computational cost and focus on the more competitive BM25 baseline.

3.4 Challenges and Solutions

The original BM25 baseline lacked preprocessing, which limited retrieval effectiveness. We addressed this by adding lowercasing, punctuation removal, and stopword filtering, which significantly improved performance. The from-scratch TF-IDF model helped us understand core IR concepts but underperformed compared to BM25, confirming the importance of robust weighting and normalization.

4 Approach 2: Representation Learning (Dense Retrieval)

4.1 Implementation

We adopt a dense retrieval approach [1] to overcome the lexical limitations of BM25 and better capture semantic similarity between tweets and scholarly papers, which often use divergent vocabulary. So our second run replaces lexical term–matching with a neural encoder that maps tweets and papers into a shared semantic space and retrieves papers by nearest-neighbour search. We started from the lightweight Sentence-BERT checkpoint all-MiniLM-L6-v2 and fine-tuned it for two epochs with the Multiple Negatives Ranking Loss provided by the sentence-transformers library. Fine-tuning triples were built by pairing each training tweet with its gold paper plus five BM25 "hard" negatives and two random negatives, for about 100k triples in total.

Each paper was represented as the concatenation title. abstract. We encoded the 7 718 papers once, applied L2 normalisation and stored the 384-d vectors in a FAISS IndexFlatIP whose inner product becomes cosine similarity after normalisation. The complete index occupies roughly 11 MB in float32.

At inference time we embed the tweet, normalise the vector and perform a single k-NN lookup (with k=10 for evaluation). The pipeline is implemented in predict_test.py; after an initial warm-up the end-to-end latency on a laptop CPU remains below 25 ms per query.

4.2 Results

Dataset	MRR@1	MRR@5	MRR@10
Train	0.561	0.639	0.639
Dev	0.532	0.599	0.599

Table 6: Train and dev sets performances of the dense retriever MiniLM.

On the training set, the dense retriever shows consistent retrieval of relevant documents within the top ranks. This marks a modest improvement over the baseline BM25.

In the development set, performance improves further, with about 4.7 percentage points higher than BM25, demonstrating stronger ranking quality beyond the top hit.

4.2.1 Evaluation on the Test Set

Dataset	MRR@1	MRR@5	MRR@10
Test	0.473	0.539	0.547

Table 7: Test set performance of the dense retriever MiniLM.

To complete the evaluation, we applied the dense retriever directly to the test set with gold annotations, without any further retraining or modification. Performance on the test set follows the same trend, confirming that the gains generalize. Nonetheless, the dense model consistently outperforms the BM25 baseline on the test set, confirming its advantage in capturing semantic similarity between short tweet queries and scholarly documents.

4.3 Challenges and Solutions

The most time-consuming step was generating hard negatives, but parallelising the BM25 pass and caching intermediate runs reduced wall-clock time from 50 min to 12 min. GPU memory was a bottleneck during fine-tuning: BERT-base exceeded the 8 GB card available in our lab, while MiniLM supported a batch size of 32 triples without gradient-checkpointing. Early experiments also revealed silent dimension mismatches between embedding sizes and the FAISS index, so we added assertions to abort if shapes diverged. Additionally, the BM25 baseline used simple whitespace tokenization without handling punctuation, casing, or multi-word expressions.

5 Approach 3: Neural Re-Ranking

We implemented two independent neural re-ranking pipelines based on Cross-Encoder architectures. While both approaches rely on the BM25 baseline for candidate selection and use a pre-trained transformer to re-score tweet–document pairs, they differ significantly in their design and training strategies.

5.1 Implementation of Zero-Shot Cross-Encoder Re-ranking

This approach uses a pre-trained Cross-Encoder model (ms-marco-MiniLM-L-6-v2) directly without fine-tuning. The Cross-Encoder architecture was selected to directly score query-article pairs based on joint input representation.

Candidate articles for re-ranking were generated by retrieving the top 10 articles from the BM25 baseline model for each query. Each query and candidate article pair was constructed by concatenating the tweet text with the article's title and abstract.

The Cross-Encoder computed a relevance score for each query-article pair. For each query, the candidate articles were sorted by these scores in descending order, and the top 5 ranked articles were selected as the final predictions.

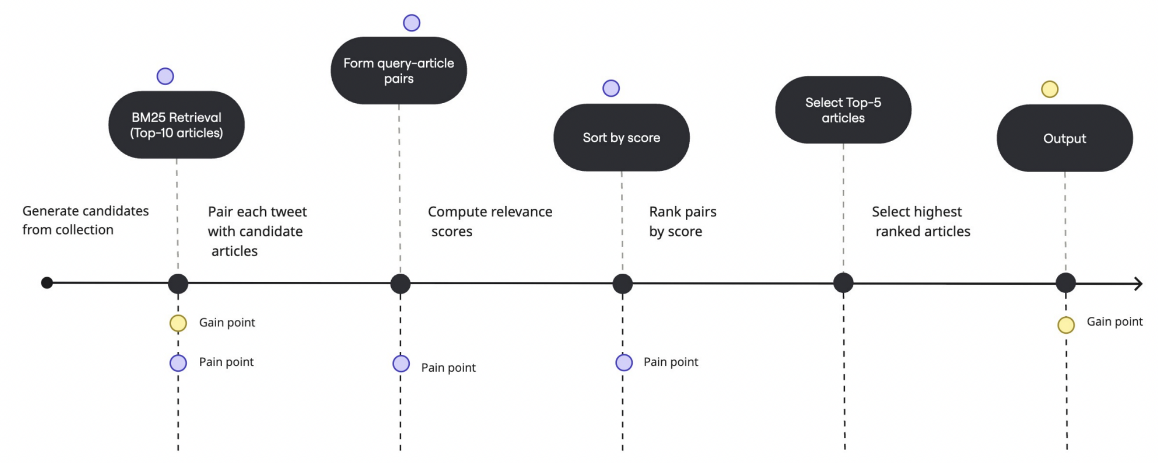

Figure 1: Zero-Shot Re-Ranking Pipeline.

The diagram uses gain points (yellow dots) and pain points (blue dots) to highlight key strengths and challenges within this method. Gain points indicate stages where performance improvements are achieved, such as initial candidate retrieval and final output. Pain points indicate stages that require significant computational resources or involve process complexity, such as pair formation, scoring, and sorting.

5.2 Fine-tuned Cross-Encoder with Pairwise Dataset

This approach extends the standard re-ranking pipeline by training a custom neural model using pairwise supervision. The system follows a two-stage architecture:

In the first stage, the BM25 algorithm was used to retrieve a large candidate set of potentially relevant articles. For training, the top 1000 BM25-ranked articles were retrieved per query to provide both relevant and hard negative examples. For evaluation, only the top 5 BM25 candidates were used to match the official metrics.

In the second stage, the candidates were re-ranked using a fine-tuned Cross-Encoder. Initially, the model was trained pointwise, but this was later replaced with a more effective pairwise training strategy, where each input consisted of a tweet paired with both a relevant and a non-relevant article. This allowed the model to learn relative relevance scores and improved its ability to distinguish between similar candidates.

For the final implementation, the pre-trained model ms-marco-MiniLM-L-6-v2 was fine-tuned using the custom pairwise dataset. Each input pair was formed by concatenating the tweet text with the article's title and abstract. During inference, the model scored each of the 5 BM25-selected candidates, which were then ranked by predicted relevance.

Unlike the zero-shot variant, this approach involved custom training and hard-negative sampling, resulting in a more specialized re-ranker tailored to the retrieval task.

5.3 Results

The fine-tuned Cross-Encoder outperforms the zero-shot approach across all MRR metrics on the development set. This confirms the benefit of task-specific training, particularly when using pairwise supervision to better differentiate relevant from non-relevant documents. However, the zero-shot model still performs competitively without any additional training.

Approach	MRR@1	MRR@5	MRR@10
Fine-Tuned Cross-Encoder	0.614	0.652	0.652
Zero-Shot Cross-Encoder	0.535	0.574	0.574

Table 8: Comparison of neural re-ranking performance on the development set.

5.3.1 Evaluation on the Test Set

Approach	MRR@1	MRR@5	MRR@10
Fine-Tuned Cross-Encoder	0.497	0.525	0.525
Zero-Shot Cross-Encoder	0.459	0.519	0.519

Table 9: Comparison of neural re-ranking performance on the test set.

The fine-tuned Cross-Encoder slightly outperforms the zero-shot approach on the test set. However, the zero-shot model achieves nearly comparable results without any task-specific training, highlighting its strength as a simple and effective re-ranking baseline.

5.4 Challenges and Solutions

Both the zero-shot and fine-tuned Cross-Encoder models incurred high computational costs during inference, as each query–document pair must be scored individually. To mitigate this, BM25 was used in both settings to pre-select a limited number of candidates: the top-10 for the zero-shot re-ranking, and top-1000 during training in the fine-tuned version (reduced to top-5 during evaluation).

Additionally, due to the Cross-Encoder's maximum token length, only the tweet, title, and the beginning of the abstract were concatenated as input, to avoid truncation and ensure input compatibility across both approaches.

In the fine-tuned Cross-Encoder, additional complexity arose from training the model itself. Initially, a pointwise dataset was used, but switching to a pairwise training strategy improved convergence and training speed.

Migrating from a generic BERT model to the more specialized ms-marco-MiniLM-L-6-v2 Cross-Encoder also required careful handling of tokenizer and model weights, but resulted in improved ranking performance.

The initial BM25 retrieval suffered from low recall due to minimal preprocessing. Refining tokenization and scoring logic helped improve the quality of candidate articles.

6 Conclusion

The BM25 baseline, while fast and interpretable, showed limited effectiveness on noisy tweet queries. The from-scratch TF-IDF model performed significantly worse, highlighting the value of optimized IR methods. Among neural approaches, the Dense Retriever (MiniLM) achieved the best overall performance on the test set (MRR@5 = 0.539), slightly outperforming the fine-tuned Cross-Encoder. This suggests that dense retrieval models can be highly competitive, especially when computational efficiency and scalability are considered. Fine-tuned Cross-Encoders still offer strong results and remain a reliable choice for high-precision re-ranking.

Model	**MRR@1**	**MRR@5**	**MRR@10**
Dense Retriever (MiniLM)	0.473	**0.539**	**0.547**
Fine-Tuned Cross-Encoder (NRR)	0.497	0.525	0.525
Zero-Shot Cross-Encoder (NRR)	0.459	0.519	0.519
BM25	0.459	0.519	0.519

Table 10: Test set performance comparison across all approaches, sorted by MRR@5

Future work could explore hybrid architectures combining sparse and dense retrieval, and further fine-tuning with domain-specific data to improve generalization and robustness.

7 References

[1] Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering," EMNLP, 2020.

8 Link to our repository in Git Hub

https://github.com/Daugon1997/AIR