Performance Comparison of Bi-Encoders and Cross-Encoders on BEIR Datasets with Fine-Tuning

CS-GY 6913 Web Search Engines Final Project

Dong Li
dl5214@nyu.edu

Dec 2024

Abstract

This paper explores the integration of bi-encoders and cross-encoders in information retrieval, highlighting their performance across four datasets: SciFact, NFCorpus, TREC-COVID, and Touche-2020. By employing a cascading query approach and utilizing HNSW for efficient retrieval, the study balances efficiency and accuracy. Fine-tuning cross-encoders on domain-specific datasets demonstrated significant improvements, particularly for SciFact, while generalization experiments revealed the critical role of dataset similarity in transferability. This work provides valuable insights into optimizing retrieval systems through neural networks and indexing methods.

Keywords: Information Retrieval, Bi-Encoder, Cross-Encoder, HNSW, Fine-Tuning

1 Introduction

Information retrieval (IR) is a fundamental task in computer science that focuses on retrieving relevant information from large corpora based on user queries. Traditional approaches to IR include **inverted indexes** [1], a sparse retrieval method that maps terms in a query to their occurrences in a corpus. While inverted indexes are highly efficient for keyword-based searches, they often struggle with semantic understanding and fail to capture relationships between terms.

In contrast, **dense retrieval** methods employ embeddings to represent queries and documents in a continuous vector space, enabling the computation of semantic similarity. Neural network-based methods, such as bi-encoders and cross-encoders, are commonly used in dense retrieval. Bi-encoders independently generate embeddings for queries and documents, offering efficiency for large-scale retrieval tasks [2, 3]. Cross-encoders, on the other hand, calculate relevance scores for query-document pairs and are often used for reranking due to their accuracy, albeit at higher computational costs [4, 5, 6].

Another important technique in IR is graph-based **approximate nearest neighbor (ANN)** search, such as the **Hierarchical Navigable Small World (HNSW)** algorithm [7]. HNSW organizes data points into layered graphs, enabling efficient nearest neighbor retrieval by navigating through a hierarchy of connections. Key parameters like `m`, `ef_construct`, and `ef_search` control the trade-off between accuracy, index size, and computational efficiency during construction and querying.

Modern IR systems often adopt a **cascading query** approach, where retrieval is divided into two stages: (1) a *coarse ranking* phase, where a bi-encoder is used to retrieve the top-k documents, and (2) a *fine ranking* phase, where a cross-encoder refines these results by reranking them based on detailed relevance scores [8, 9]. This cascading architecture effectively balances efficiency and accuracy, leveraging the strengths of both retrieval and reranking models.

The remainder of this paper is organized as follows: Section 2 introduces the overall structure and workflow of the information retrieval system, including the bi-encoder and cross-encoder models used.

Section 3 compares the performance of these models on various datasets. Section 4 describes the fine-tuning process for cross-encoders on dataset-specific training data and evaluates their generalization to other datasets. Section 5 discusses the strengths, limitations, and potential future directions of this work. Finally, Section 6 provides a summary of findings and conclusions.

2 System Structures

2.1 Overall Structure

Figure 1 illustrates the overall structure of the information retrieval system. The process begins with a bi-encoder that independently encodes the corpus and queries into dense vector representations. These embeddings allow for similarity calculation between queries and documents, typically using cosine similarity or reciprocal distance. For each query, we compute its similarity with all corpus embeddings to retrieve the top-k results.
To handle multiple queries efficiently, an index structure is utilized for faster retrieval. Specifically, a Hierarchical Navigable Small World (HNSW) index is constructed with parameters m=15 , ef_construct=300 , and ef_search=1000 . While HNSW is an approximate nearest neighbor (ANN) approach, as shown in Section 2.2.4, its performance is comparable to the DenseRetrievalExactSearch method provided by the BEIR package. Therefore, HNSW is used to accelerate the search process, enabling the retrieval of the top 300 documents for each query.
In the next stage, a cross-encoder calculates a relevance score for each query-document pair based on the query embeddings and corresponding retrieved documents. These scores are used to rerank the documents. Finally, the top 100 documents are selected based on the cross-encoder scores and used as the final search results for evaluation. This pipeline combines the efficiency of bi-encoders with the accuracy of cross-encoders to deliver high-quality retrieval results.

2.2 Dataset and Model Selection

2.2.1 GPU Resources

The experiments were conducted on a system featuring an NVIDIA GeForce GTX 1080 Ti GPU with 11GB of VRAM, running on NVIDIA driver version 530.30.02 and CUDA version 11.7. This setup provided sufficient computational power for training and evaluating neural models in various information retrieval tasks.

2.2.2 Datasets

The evaluation was performed using the pytrec_eval package [10, 11] on four BEIR datasets [12, 13], each selected for its unique characteristics and relevance proportions. The datasets included:
- **SciFact**: A scientific claim verification dataset with a relevance D/Q (relevant documents per query) value of 1.1, focusing on retrieving evidence to support or refute claims.
- **NFCorpus**: A biomedical dataset with a relevance D/Q value of 38.2, emphasizing fine-grained relevance for medical literature.
- **TREC-COVID**: A dataset addressing information needs related to COVID-19 research and medical queries, with a relevance D/Q value of 493.5.
- **Touche-2020**: An argumentative retrieval dataset designed for tasks involving debates and opinionated content, with a relevance D/Q value of 19.0.
These datasets were chosen to evaluate the generalizability and effectiveness of different models across diverse domains and tasks.

2.2.3 Bi-Encoder Models

For bi-encoders, I used two models for comparison:

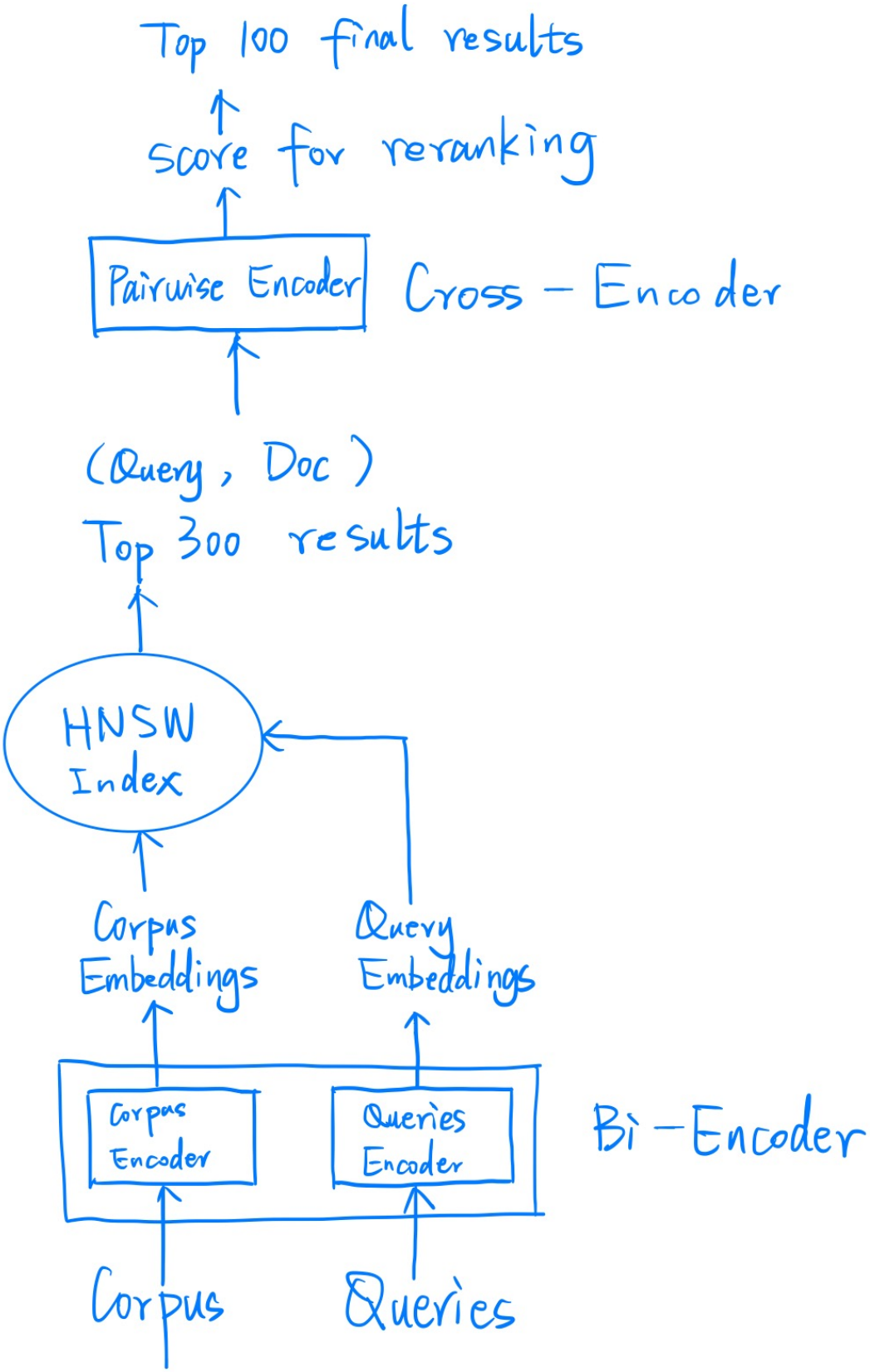

Figure 1: Schematic of the information retrieval system. The system encodes the corpus and queries using a bi-encoder to produce embeddings, which are indexed with HNSW for efficient retrieval. Query embeddings are matched to retrieve the top 300 results, which are reranked by a cross-encoder based on relevance scores. The final output consists of the top 100 ranked results per query for evaluation.

- **TAS-B** (sentence-transformers/msmarco-distilbert-base-tas-b) [2], optimized for MS MARCO and performing well on small to medium-scale datasets requiring accurate ranking.
- **E5** (intfloat/e5-base-v2) [3], designed for general-purpose text embeddings and excelling in semantic search and multilingual applications.

2.2.4 HNSW Index

Rather than computing the pairwise distance (or cosine similarity) between a query and all corpus embeddings, I utilized an HNSW index with parameters m=15, ef_construct=300, and ef_search=1000.

Table 1: Comparison of Dense Retrieval Exact Search and HNSW ANN Methods

Method	Time (min:sec)	MAP	NDCG@10	NDCG@30	Recall@10	Recall@30
Exact Search	03:23	0.6836	0.7210	0.7378	0.8419	0.9070
HNSW ANN	<02:00	0.6701	0.7121	0.7262	0.8464	0.9003

Table 1 presents the performance of the E5 bi-encoder on the SciFact dataset without reranking. The results demonstrate that the HNSW method performs comparably to exact search, with only a minimal decrease in performance metrics, while requiring significantly less time. Therefore, for efficiency considerations, this project adopts the HNSW index for all subsequent comparisons.

2.2.5 Cross-Encoder Models

For cross-encoders, I utilized three pre-trained models:
- **Tiny-BERT** (cross-encoder/ms-marco-TinyBERT-L-2-v2) [4], a compact model suitable for efficient, resource-constrained environments.
- **MiniLM** (cross-encoder/ms-marco-MiniLM-L-6-v2) [5], balancing size and performance for tasks requiring speed and accuracy.
- **Electra** (cross-encoder/ms-marco-electra-base) [6], a larger model designed for robust performance, particularly for complex queries.

All three cross-encoders were pre-trained on the MS MARCO dataset [14], a large-scale benchmark for information retrieval tasks.
To further enhance performance on domain-specific tasks, I fine-tuned two cross-encoders (Tiny-BERT and MiniLM) on the training sets of **SciFact** and **NFCorpus**. This fine-tuning resulted in a total of seven cross-encoders: three pre-trained models and four fine-tuned models (two models fine-tuned on each dataset). Fine-tuning allowed these cross-encoders to adapt to specific dataset characteristics, such as scientific claim verification in SciFact and granular biomedical relevance in NFCorpus, thereby optimizing their retrieval and reranking capabilities.

3 Comparison of Bi-Encoders and Cross-Encoders

This section evaluates the performance of TAS-B (sentence-transformers/msmarco-distilbert-base-tas-b) [2] and E5 (intfloat/e5-base-v2) [3] bi-encoders, alongside Tiny-BERT (cross-encoder/ms-marco-TinyBERT-L-2-v2) [4], MiniLM (cross-encoder/ms-marco-MiniLM-L-6-v2) [5], and Electra (cross-encoder/ms-marco-electra-base) [6] cross-encoders. The comparison spans four BEIR datasets [12]: SciFact, NFCorpus, TREC-COVID, and Touche-2020, using evaluation metrics defined in Homework 3 Section 4.2. The results are presented in Table 2.

3.1 Comparison of Bi-Encoders on Different Datasets without Reranking

Figure 2 compares the performance of two bi-encoders, TAS-B and E5, across various metrics on four datasets. Precision metrics are excluded, as queries in some datasets may have only a few relevant

Table 2: Experimental Results Summary (Without Fine-Tuning). Abbreviations: DS = Dataset (SF = SciFact, NF = NFCORPUS, TC = TREC-COVID, T20 = TOUCHE2020), BE = Bi-Encoder (TB = TAS-B, E5 = E5), CE = Cross-Encoder (TB = Tiny-BERT, ML = Mini-LM, EL = Electra, None = No Cross-Encoder), N10 = NDCG@10, N30 = NDCG@30, R10 = Recall@10, R30 = Recall@30, P10 = Precision@10, P30 = Precision@30.

DS	BE	CE	MAP	MRR	N10	N30	R10	R30	P10	P30
SF	TB	None	0.4611	0.4786	0.5048	0.5311	0.6500	0.7509	0.0730	0.0286
SF	TB	TB	0.5720	0.5824	0.6128	0.6310	0.7457	0.8180	0.0840	0.0310
SF	TB	ML	0.6016	0.6154	0.6423	0.6600	0.7689	0.8397	0.0863	0.0317
SF	TB	EL	0.5996	0.6123	0.6382	0.6576	0.7586	0.8403	0.0867	0.0318
SF	E5	None	0.6704	0.6779	0.7121	0.7262	0.8464	0.9003	0.0950	0.0341
SF	E5	TB	0.5999	0.6086	0.6438	0.6631	0.7932	0.8697	0.0890	0.0328
SF	E5	ML	0.6280	0.6396	0.6697	0.6924	0.8056	0.8969	0.0903	0.0337
SF	E5	EL	0.6087	0.6181	0.6463	0.6678	0.7718	0.8587	0.0877	0.0326
NF	TB	None	0.1132	0.4441	0.2564	0.2285	0.1197	0.1678	0.1938	0.1134
NF	TB	TB	0.1489	0.5463	0.3268	0.2921	0.1598	0.2050	0.2347	0.1411
NF	TB	ML	0.1486	0.5546	0.3338	0.2931	0.1555	0.1967	0.2378	0.1404
NF	TB	EL	0.1223	0.4823	0.2873	0.2515	0.1363	0.1785	0.2130	0.1200
NF	E5	None	0.1676	0.5532	0.3431	0.3131	0.1665	0.2292	0.2560	0.1563
NF	E5	TB	0.1489	0.5463	0.3268	0.2921	0.1598	0.2050	0.2347	0.1411
NF	E5	ML	0.1570	0.5684	0.3405	0.3060	0.1600	0.2095	0.2418	0.1479
NF	E5	EL	0.1277	0.5036	0.2903	0.2613	0.1396	0.1867	0.2099	0.1254
TC	TB	None	0.0403	0.5110	0.3164	0.2771	0.0079	0.0194	0.3440	0.2953
TC	TB	TB	0.0499	0.8733	0.5674	0.4992	0.0159	0.0374	0.6200	0.5333
TC	TB	ML	0.0543	0.7857	0.5961	0.5187	0.0163	0.0389	0.6760	0.5567
TC	TB	EL	0.0477	0.7725	0.5300	0.4737	0.0147	0.0368	0.5980	0.5160
TC	E5	None	0.0295	0.1946	0.1393	0.1319	0.0038	0.0107	0.1560	0.1473
TC	E5	TB	0.0000	0.0142	0.0045	0.0022	0.0002	0.0002	0.0080	0.0027
TC	E5	ML	0.0154	0.2055	0.1429	0.1286	0.0041	0.0101	0.1660	0.1440
TC	E5	EL	0.0129	0.1896	0.1230	0.1141	0.0034	0.0094	0.1380	0.1293
T20	TB	None	0.1296	0.4484	0.2012	0.2279	0.1378	0.2399	0.1837	0.1177
T20	TB	TB	0.1579	0.5589	0.2937	0.2954	0.1802	0.2763	0.2571	0.1388
T20	TB	ML	0.1927	0.5999	0.3149	0.3101	0.1914	0.3198	0.2857	0.1660
T20	TB	EL	0.1954	0.6641	0.3437	0.3407	0.2150	0.3232	0.3102	0.1707
T20	E5	None	0.1746	0.5491	0.2660	0.2976	0.1578	0.2876	0.2224	0.1463
T20	E5	TB	0.1848	0.5715	0.2998	0.3222	0.1908	0.3201	0.2735	0.1639
T20	E5	ML	0.2091	0.5989	0.3268	0.3441	0.2065	0.3369	0.3000	0.1714
T20	E5	EL	0.2100	0.6749	0.3506	0.3553	0.2231	0.3380	0.3184	0.1741

documents.

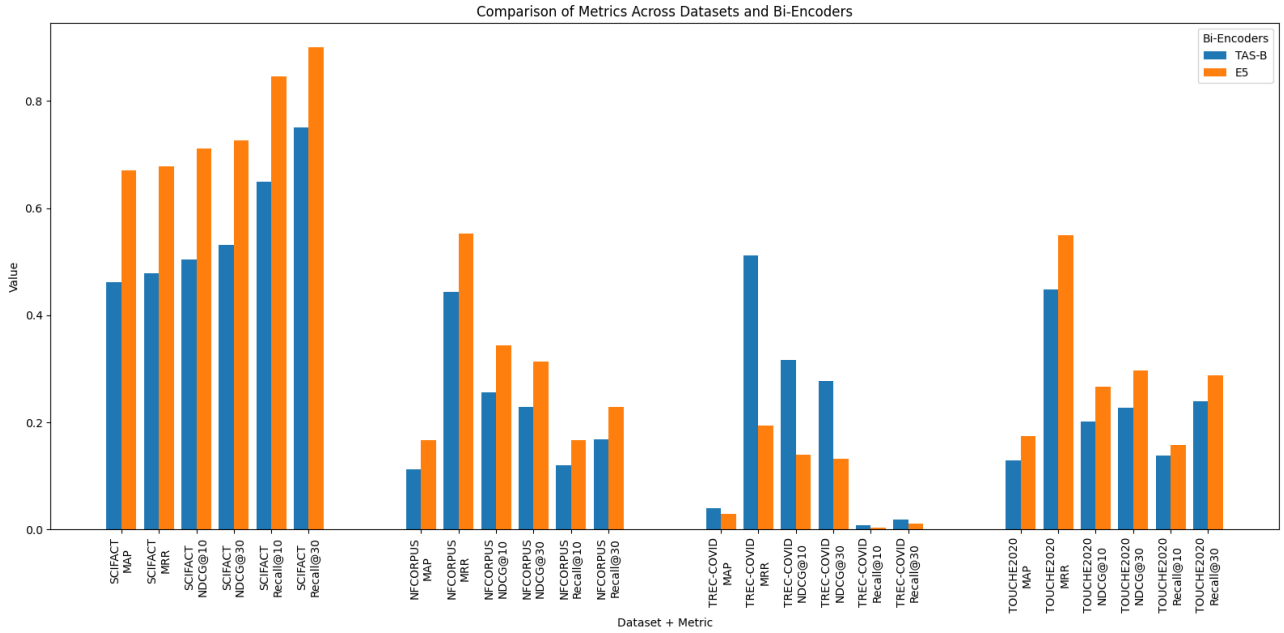

Figure 2: Performance comparison of TAS-B and E5 bi-encoders on four datasets (SciFact, NFCorpus, TREC-COVID, and Touche-2020) across various metrics, including MAP, MRR, NDCG@10, NDCG@30, Recall@10, and Recall@30. Each group of bars represents the results for a specific metric, with TAS-B shown in blue and E5 in orange for comparison.

From Figure 2, E5 demonstrates superior performance over TAS-B in most cases, achieving higher scores across metrics. However, on the TREC-COVID dataset, TAS-B significantly outperforms E5. This could be attributed to TREC-COVID's high Rel D/Q rate (493.5), where TAS-B's focus on query-answer tasks proves advantageous. In contrast, E5, being a general-purpose model, excels in broader applications but may struggle with nuanced differences or specialized fields.
In cascading retrieval setups, bi-encoders perform initial retrieval, with cross-encoders refining the results. TAS-B remains a viable choice due to its faster speed compared to E5 and its superior performance on datasets like TREC-COVID, where domain-specific relevance is critical.

3.2 Comparison of Cross-Encoders' Reranking Performance

Figures 3,4,5, and 6 present the performance of different bi-encoder and cross-encoder combinations on the SciFact, NFCorpus, TREC-COVID, and Touche-2020 datasets, respectively. All cross-encoders were pre-trained on the MS MARCO dataset, which can influence their performance. Specifically, datasets that diverge significantly from MS MARCO in terms of structure, content, or domain characteristics may see diminished benefits from cross-encoders, and in some cases, the performance may even lag behind models like E5 bi-encoder alone.
In Figures 3 and 4, E5 bi-encoder demonstrates remarkable performance, often surpassing the results obtained with cross-encoder reranking. This suggests that using only the embeddings generated by E5 is not only more efficient but can also yield better results than incorporating cross-encoders. For TAS-B, however, all three cross-encoders significantly enhance the performance through reranking. Among the cross-encoders, MiniLM provides the best reranking results for SciFact and NFCorpus datasets, while Electra—despite being the largest model—performs relatively poorly.
In Figure 5, we observe the reranking performance on the TREC-COVID dataset. As discussed earlier, the E5 bi-encoder struggles with the initial retrieval on this dataset. Hence, the focus shifts to reranking the top 300 results from TAS-B bi-encoder. All three cross-encoders significantly improve

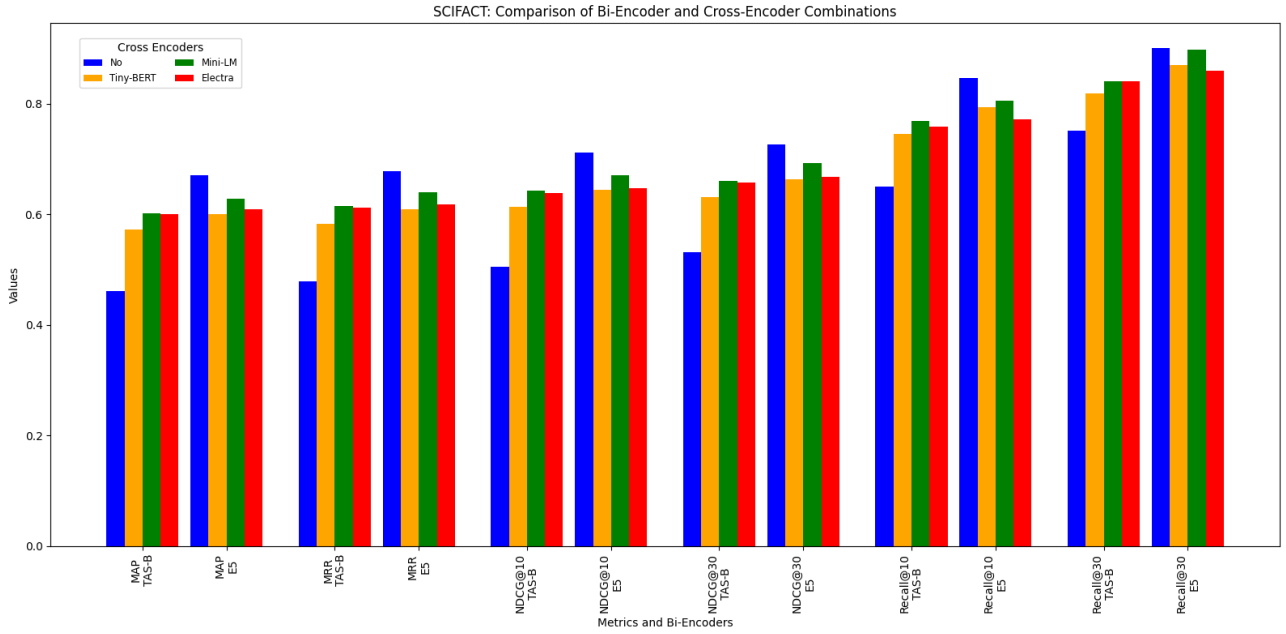

Figure 3: Performance comparison of TAS-B and E5 bi-encoders combined with cross-encoders (Tiny-BERT, MiniLM, Electra) on the SciFact dataset across various metrics. For each group of bars, the columns from left to right represent: no cross-encoder, Tiny-BERT, MiniLM, and Electra.

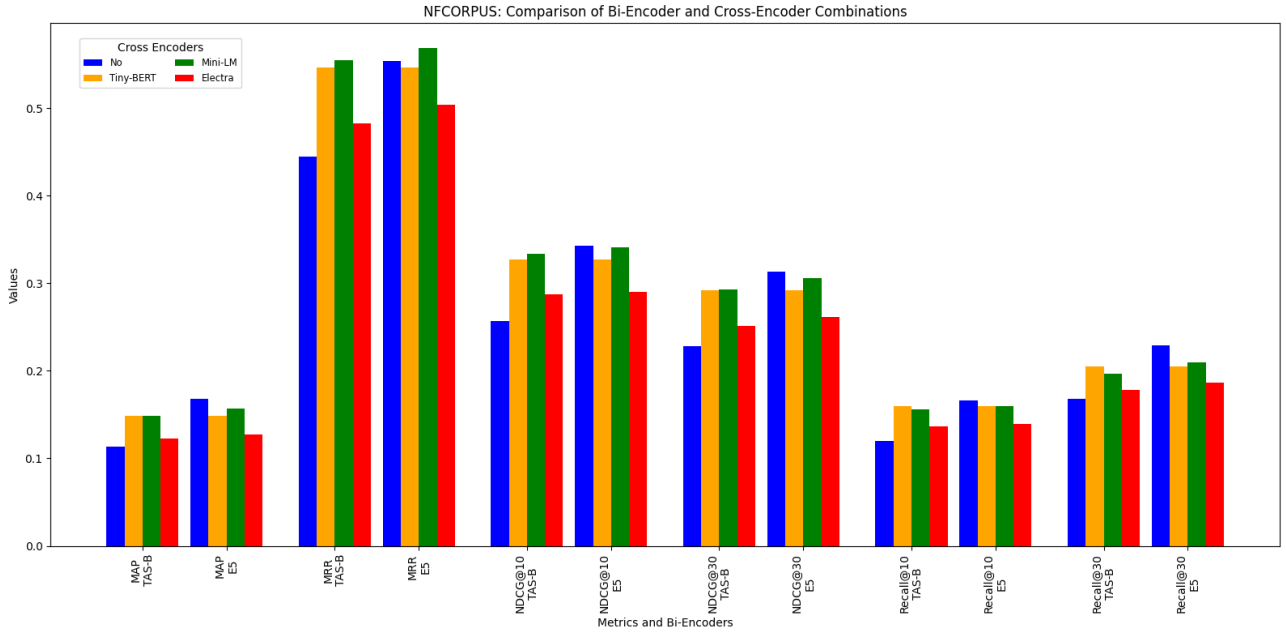

Figure 4: Performance comparison of TAS-B and E5 bi-encoders combined with cross-encoders (Tiny-BERT, MiniLM, Electra) on the NFCorpus dataset across various metrics. For each group of bars, the columns from left to right represent: no cross-encoder, Tiny-BERT, MiniLM, and Electra.

the performance, with MiniLM consistently delivering the best results and Electra performing the worst.

Figure 6 highlights the performance on the Touche-2020 dataset. The expected trend is observed: performance improves progressively from using no cross-encoder, to Tiny-BERT, then MiniLM, and finally Electra, reflecting the increasing size and complexity of these models. However, this pattern is not consistent across all datasets, as MiniLM or even the E5 bi-encoder alone often delivers better

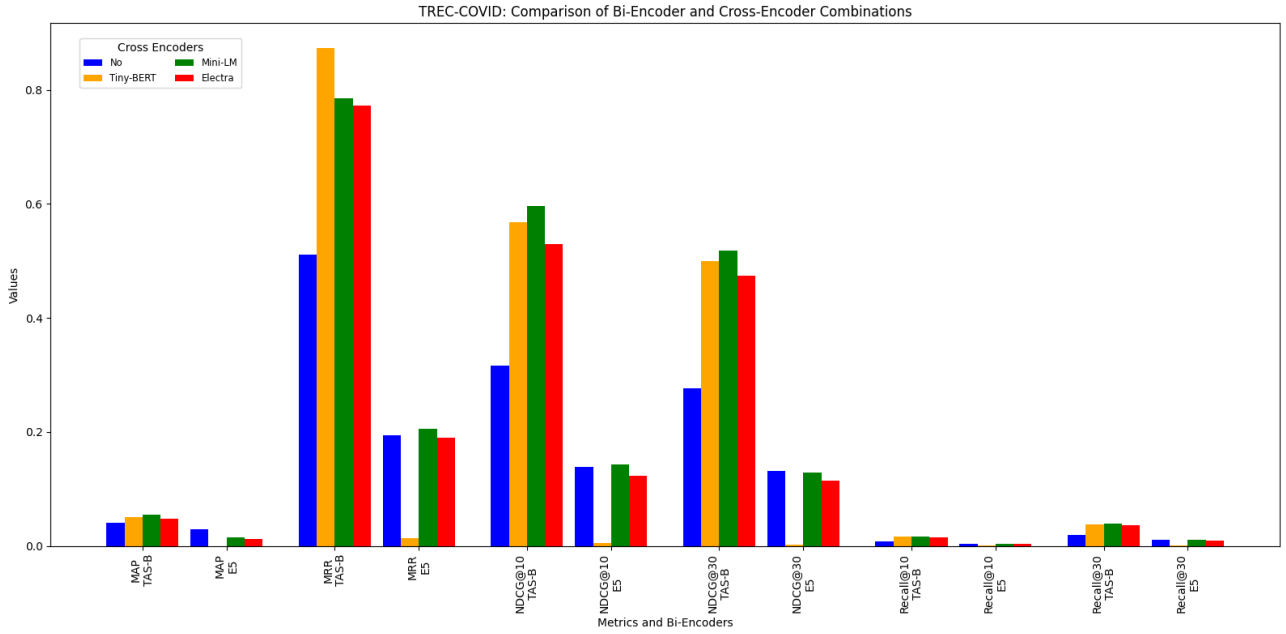

Figure 5: Performance comparison of TAS-B and E5 bi-encoders combined with cross-encoders (Tiny-BERT, MiniLM, Electra) on the TREC-COVID dataset across various metrics. For each group of bars, the columns from left to right represent: no cross-encoder, Tiny-BERT, MiniLM, and Electra.

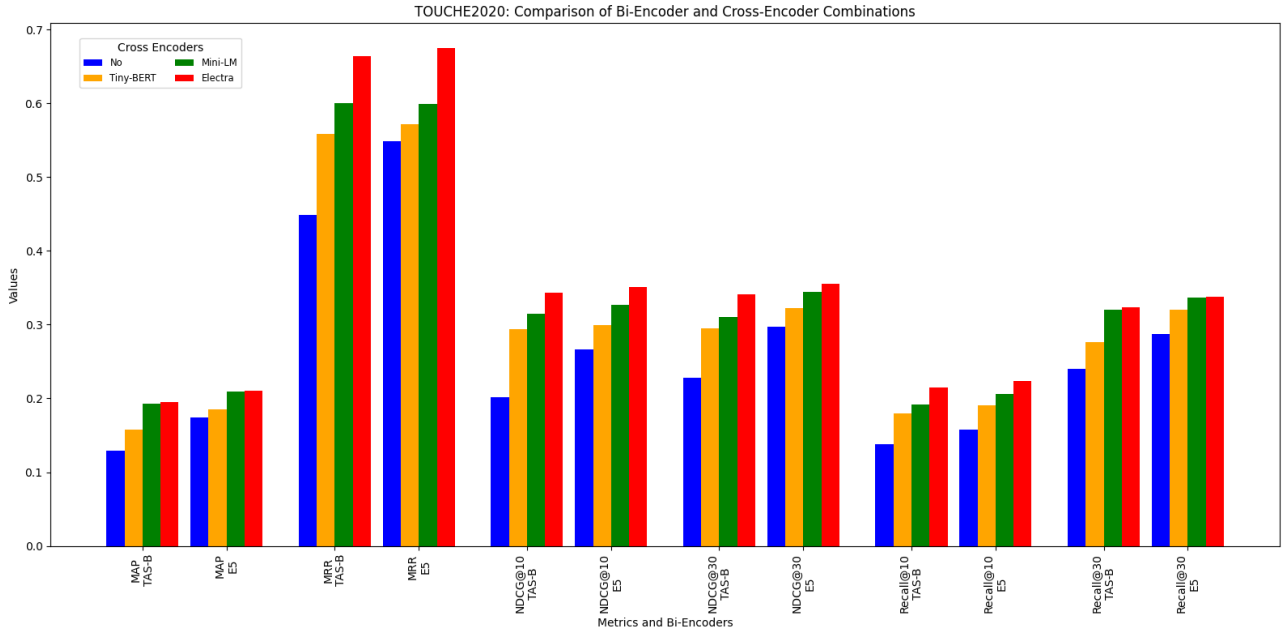

Figure 6: Performance comparison of TAS-B and E5 bi-encoders combined with cross-encoders (Tiny-BERT, MiniLM, Electra) on the Touche-2020 dataset across various metrics. For each group of bars, the columns from left to right represent: no cross-encoder, Tiny-BERT, MiniLM, and Electra.

performance compared to other methods on SciFact, NFCorpus, and TREC-COVID datasets.
These varying trends can be attributed to several factors. Dataset characteristics, such as relevance document-to-query (D/Q) ratio, play a crucial role; for example, datasets with higher D/Q ratios might benefit more from reranking. Additionally, the similarity of the dataset to MS MARCO, the pretraining corpus, can influence cross-encoder performance. Electra's underperformance, despite its model size, might stem from overfitting to MS MARCO or suboptimal adaptation to the nuances of

specific datasets. Conversely, MiniLM's consistent performance suggests that smaller, well-optimized models may generalize better across diverse datasets. Lastly, the domain-specific nature and structural differences of the datasets may impact how effectively cross-encoders can refine the initial retrieval results. This highlights the importance of selecting models tailored to the dataset's characteristics rather than assuming that larger models will always yield better results.

4 Fine-Tuning Cross-Encoders and Evaluating Generalization Performance

This section explores the fine-tuning of Tiny-BERT and MiniLM cross-encoders using the training sets of the SciFact and NFCorpus datasets. Additionally, their generalization performance is evaluated on TREC-COVID and Touche-2020 datasets.

4.1 Fine-Tuning Cross-Encoders

4.1.1 Fine-Tuning on the SciFact Dataset

The SciFact training set provides binary relevance labels with 919 positive (label 1) samples. Since the dataset contains significantly more negative samples than positive, a negative sampling ratio of 10:1 was applied. For each query, 10 irrelevant documents were randomly selected as negative samples to ensure the training set was balanced while maintaining a larger proportion of irrelevant samples, which is critical for effective training.
The original cross-encoder outputs scores for reranking, whereas the training set uses binary labels (0 and 1). To align with this, a sigmoid function was applied to the cross-encoder's output scores (typically ranging between -12 and 8) to map them into a probability-like range. The loss function used was the default implementation in the SentenceTransformers CrossEncoder package.
Fine-tuning was performed using the following parameters:
- **Epochs**: 3
- **Batch size**: 16
- **Learning rate**: 2×10^{-5}
- **Warmup ratio**: 0.1

Both Tiny-BERT and MiniLM cross-encoders were fine-tuned using these settings. After fine-tuning, reranking was performed with the updated models. The results are presented in Figure 7.
From Figure 7, two key observations emerge:
1. Fine-tuned cross-encoders outperform their non-fine-tuned counterparts on the SciFact dataset.
2. While the non-fine-tuned MiniLM cross-encoder performs worse than the E5 bi-encoder, the fine-tuned MiniLM cross-encoder achieves better performance than E5.

These results demonstrate that cross-encoders are effective for reranking and that fine-tuning on the specific dataset significantly improves performance. Fine-tuned models, such as MiniLM, provide more reliable results and highlight the potential of dataset-specific adaptation.

4.1.2 Fine-Tuning on the NFCorpus Dataset

The training parameters are consistent with Section 4.1.1, except for the dataset specifics. The NFCorpus training set includes 110,575 positive samples (label 1), complemented with 938,541 negative samples to balance the dataset. Training this model required over 10 hours. To explore the impact of dataset size on performance and mitigate potential overfitting, I also fine-tuned the Tiny-BERT and MiniLM cross-encoders on a smaller subset consisting of 2,000 positive samples and 20,000 negative samples. The results are summarized in Figure 8.
From Figure 8, the following observations can be made:

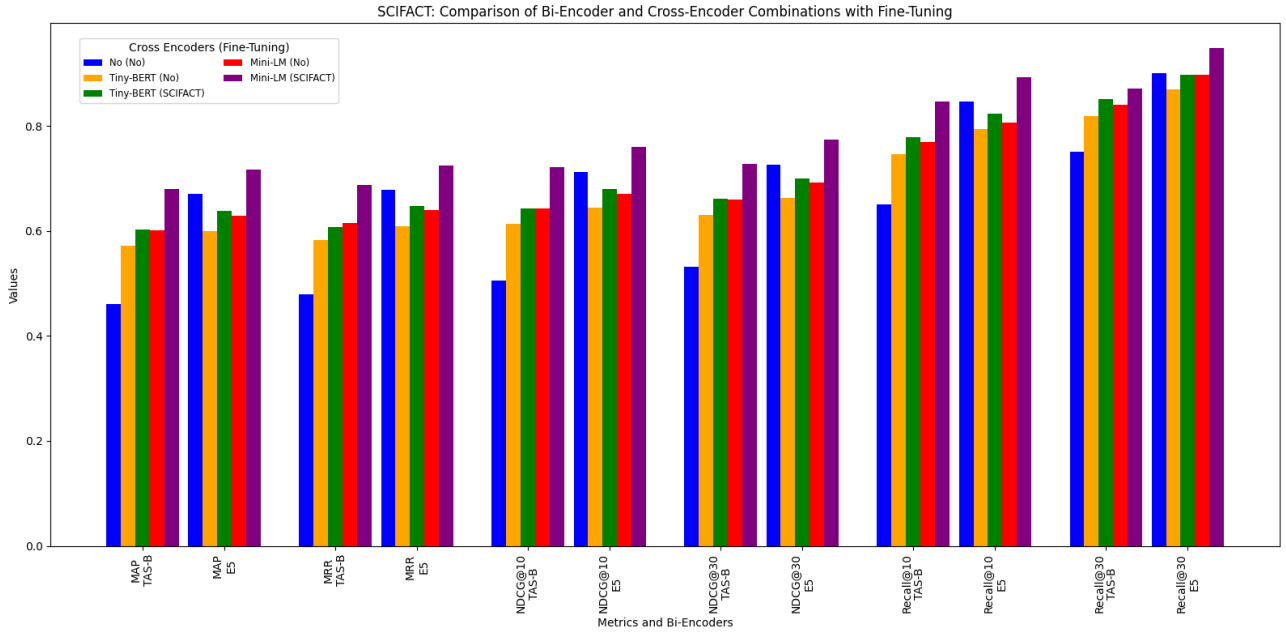

Figure 7: Comparison of reranking performance on the SciFact dataset for Tiny-BERT and MiniLM cross-encoders, before and after fine-tuning. For each group of bars, the columns from left to right represent: no cross-encoder, Tiny-BERT without fine-tuning, fine-tuned Tiny-BERT, MiniLM without fine-tuning, and fine-tuned MiniLM.

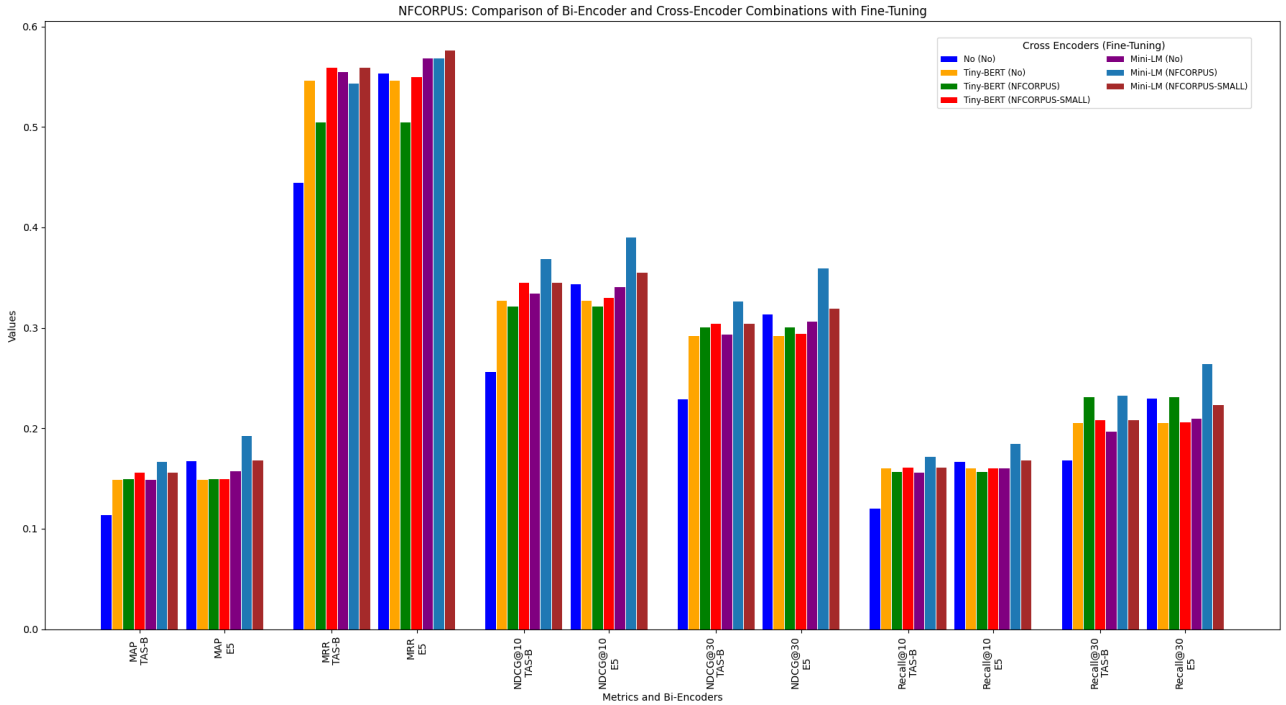

Figure 8: Comparison of cross-encoder performance on NFCorpus after fine-tuning with the full dataset and a smaller subset. Metrics include MAP, MRR, NDCG@10, NDCG@30, Recall@10, and Recall@30. For each group of bars, the columns from left to right represent: no cross-encoder, Tiny-BERT without fine-tuning, Tiny-BERT fine-tuned using the full training set, Tiny-BERT fine-tuned using a subset of the training set, MiniLM without fine-tuning, MiniLM fine-tuned using the full training set, and MiniLM fine-tuned using a subset of the training set.

1. Fine-tuning the **Tiny-BERT** cross-encoder sometimes leads to reduced performance in metrics such as MRR, NDCG@10, and Recall@10, likely due to overfitting. However, using a smaller subset for fine-tuning improves performance, suggesting that a reduced dataset size can help mitigate overfitting.
2. Fine-tuning the **MiniLM** cross-encoder improves its performance across almost all metrics. However, using the smaller subset does not surpass the performance achieved with the full dataset, although it still outperforms the non-fine-tuned version.

These results highlight the trade-off between training dataset size and model performance. While larger datasets generally improve generalization, excessive data can lead to overfitting, especially for models like Tiny-BERT. On the other hand, MiniLM benefits from the full dataset, demonstrating its capacity to leverage larger training data effectively. This suggests that optimal dataset size may vary depending on the model architecture and complexity. Moreover, fine-tuning consistently enhances performance compared to non-fine-tuned cross-encoders, reaffirming the importance of adapting pre-trained models to specific datasets.

4.2 Evaluating the Generalization Performance on TREC-COVID and Touche-2020 Datasets

From Section 4.1, we observed improved performance when cross-encoders were fine-tuned on specific datasets. This section evaluates the generalization capabilities of these fine-tuned cross-encoders on other datasets. For clarity, only cross-encoders fine-tuned on the complete SciFact and NFCorpus training sets are evaluated, excluding those fine-tuned on NFCorpus subsets.

Figure 9 presents the evaluation results on the TREC-COVID dataset. As discussed earlier, the preliminary top-300 results retrieved by the E5 bi-encoder on this dataset were suboptimal. While the reranking results for E5 are included in the figure for completeness, they are not emphasized in the analysis, as reranking predominantly irrelevant documents is not meaningful. Figure 10 displays the evaluation results on the Touche-2020 dataset.

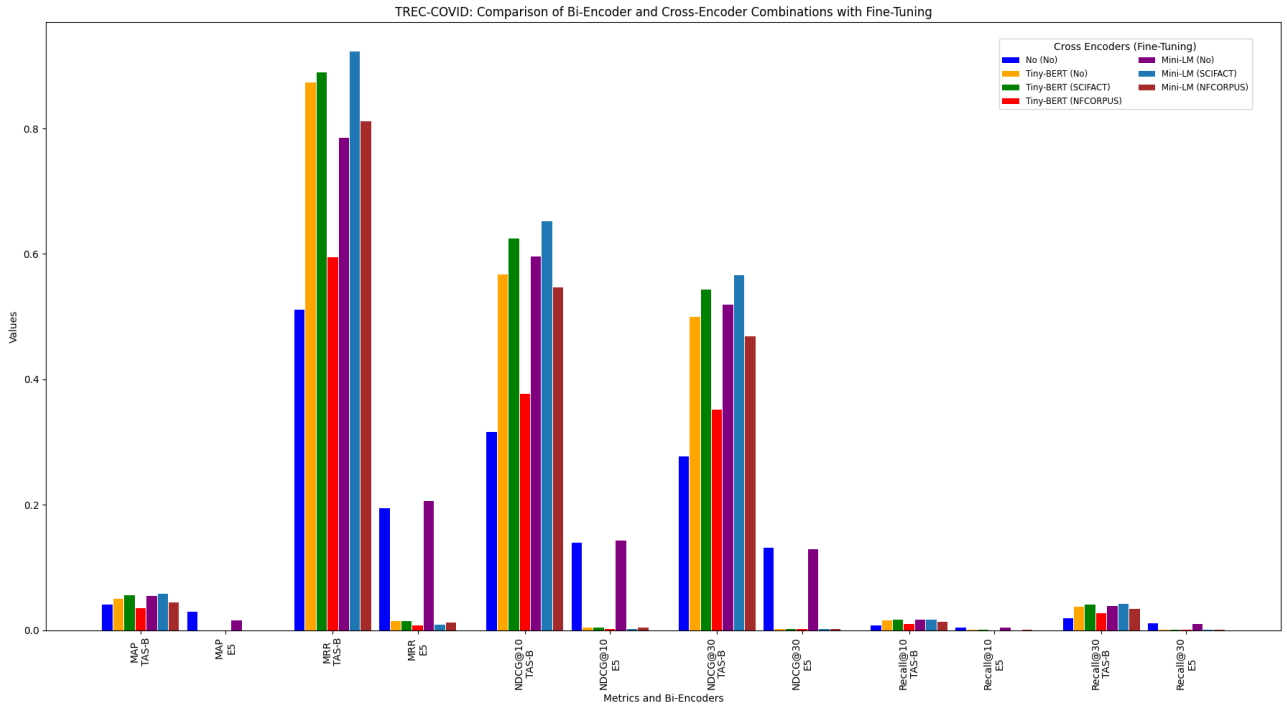

Figure 9: Generalization performance on the TREC-COVID dataset. From left to right: no cross-encoder, Tiny-BERT (no fine-tuning), Tiny-BERT (fine-tuned on SciFact), Tiny-BERT (fine-tuned on NFCorpus), MiniLM (no fine-tuning), MiniLM (fine-tuned on SciFact), and MiniLM (fine-tuned on NFCorpus).

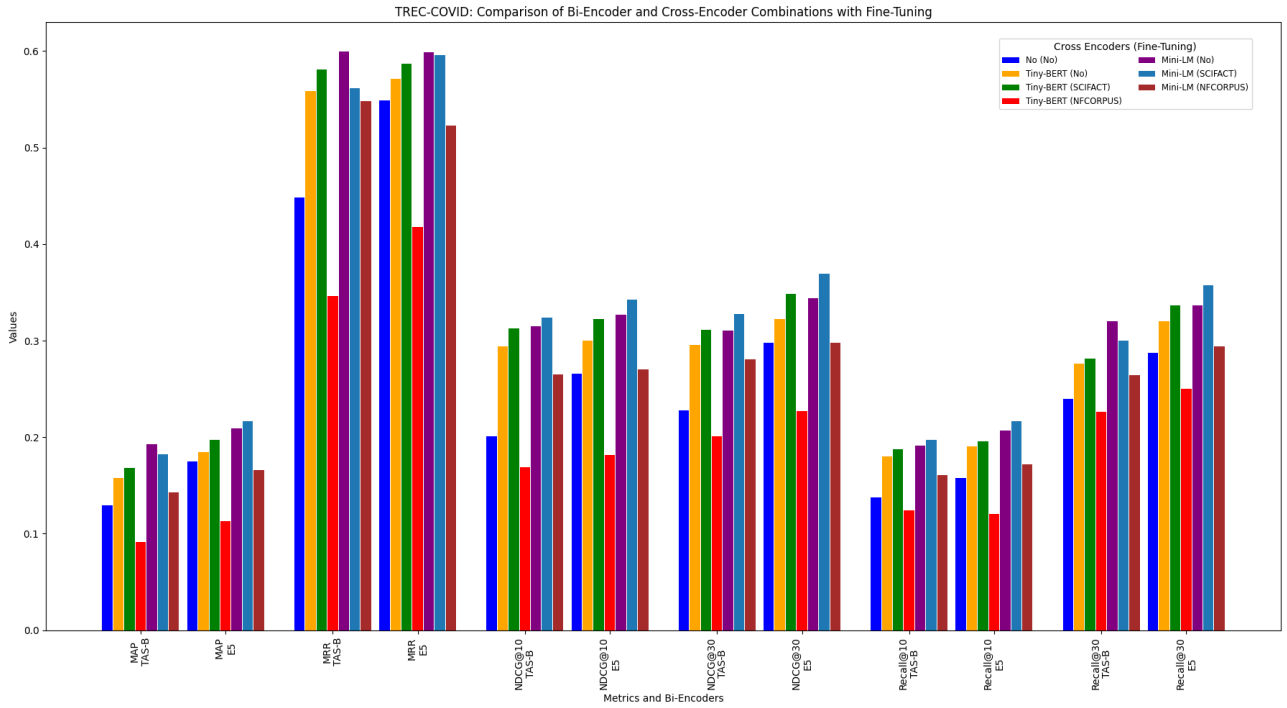

Figure 10: Generalization performance on the Touche-2020 dataset. From left to right: no cross-encoder, Tiny-BERT (no fine-tuning), Tiny-BERT (fine-tuned on SciFact), Tiny-BERT (fine-tuned on NFCorpus), MiniLM (no fine-tuning), MiniLM (fine-tuned on SciFact), and MiniLM (fine-tuned on NFCorpus).

From Figures 9 and 10, we observe the following trends:
1. Cross-encoders fine-tuned on the SciFact training set consistently outperform their non-fine-tuned counterparts on both TREC-COVID and Touche-2020 datasets.
2. Conversely, cross-encoders fine-tuned on NFCorpus generally underperform compared to their non-fine-tuned versions.
These findings indicate that fine-tuning on SciFact enhances generalization performance on TREC-COVID and Touche-2020, while fine-tuning on NFCorpus exhibits limited transferability. This disparity may be attributed to differences in dataset characteristics. SciFact's focus on scientific claims aligns better with the query-document relationship in TREC-COVID and Touche-2020, facilitating effective generalization. In contrast, NFCorpus, primarily targeting niche biomedical queries, may introduce overfitting or domain-specific biases that hinder performance on datasets with different content and structures. This underscores the importance of selecting a fine-tuning dataset that closely aligns with the target dataset in terms of structure, domain, and relevance distribution to achieve optimal generalization.

5 Discussion

5.1 Strengths

This work involved substantial effort and detailed analysis, aiming to address challenges in information retrieval with state-of-the-art techniques. By integrating tools like HNSW ANN for efficient retrieval, it showcases a practical application of advanced methodologies learned in class to tackle real-world problems. The incorporation of pre-trained models, complemented by fine-tuning on domain-specific datasets, demonstrates an effective use of neural networks and high-performance computing resources. Despite the reliance on pre-trained models, the fine-tuning experiments highlight an effort to adapt these models to new tasks and improve their utility for specific datasets.

5.2 Limitations

Several limitations remain in this study. First, the training parameters were not extensively tuned, and computational resources constrained the scope of experimentation, including the inability to evaluate on larger datasets like MS MARCO. Furthermore, while the work showcases achieved performance, it does not rigorously compare these results to benchmarks reported in the original papers of the pre-trained models.
Future work could address these gaps by exploring dynamic strategies for optimizing dataset size and balancing computational efficiency with generalization. Additionally, deeper investigations into dataset similarity metrics and their influence on cross-domain performance could inform better fine-tuning strategies. Examining the pretraining approaches of models and their adaptability across diverse domains could also provide valuable insights for improving retrieval systems.

6 Conclusion

This study presents a comprehensive exploration of information retrieval using state-of-the-art bi-encoders and cross-encoders across multiple datasets. The performance comparison highlights the strengths and limitations of different models, emphasizing the importance of selecting appropriate methods for domain-specific tasks.
Fine-tuning cross-encoders on domain-specific datasets showed promising results, particularly for SciFact, where fine-tuned models consistently outperformed their non-fine-tuned counterparts. However, generalization experiments revealed that the transferability of fine-tuned models varies depending on the similarity between the training and evaluation datasets. This underscores the importance of understanding dataset characteristics and aligning model adaptations with the target task.
Despite limitations such as restricted computational resources and the lack of evaluation on larger benchmarks like MS MARCO, this study provides valuable insights into the trade-offs between efficiency and accuracy in information retrieval. Future work could explore dynamic strategies for optimizing training parameters and dataset selection, as well as investigating the impact of pretraining strategies on cross-domain performance. Overall, this work demonstrates the potential of combining bi-encoders, cross-encoders, and efficient indexing methods to achieve high-quality and scalable retrieval results.

Appendix

The original and complete data containing detailed results, including metrics such as MAP, MRR, NDCG@k, Recall@k, and P@k, can be accessed here.

Acknowledgement

I would like to thank Prof. Torsten Suel for his excellent lectures, insightful discussions, and timely feedback on my project proposal. I also appreciate the TAs, Mehran Banka and Jinrui Gou, for their hard work and prompt feedback on assignments.

Declaration

This project, including the code and report, was completed independently. However, general ideas, such as the scope, **system structures, dataset and model selection**, and GPU usage, were discussed with **Zheng-Chen Yao (zy2876@nyu.edu)**. While these discussions were helpful, all implementations and the final report were completed **individually**.

References

[1] Justin Zobel and Alistair Moffat. "Inverted files for text search engines". In: *ACM computing surveys (CSUR)* 38.2 (2006), 6–es.

[2] Sebastian Hofstätter et al. "Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling". In: *Proc. of SIGIR*. 2021.

[3] Liang Wang et al. "Text Embeddings by Weakly-Supervised Contrastive Pre-training". In: *arXiv preprint arXiv:2212.03533* (2022).

[4] Xiaoqi Jiao et al. "Tinybert: Distilling bert for natural language understanding". In: *arXiv preprint arXiv:1909.10351* (2019).

[5] Wenhui Wang et al. "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

[6] Kevin Clark et al. "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators". In: *International Conference on Learning Representations (ICLR)*. 2020.

[7] Yu A Malkov and Dmitry A Yashunin. "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs". In: *IEEE transactions on pattern analysis and machine intelligence* 42.4 (2018), pp. 824–836.

[8] Rodrigo Nogueira and Kyunghyun Cho. "Passage Re-ranking with BERT". In: *arXiv preprint arXiv:1901.04085* (2019).

[9] Sebastian Hofstätter et al. "Efficiently teaching an effective dense retriever with balanced topic aware sampling". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 113–122.

[10] EM Voorhees. *TREC: Experiment and evaluation in information retrieval*. 2005.

[11] Christophe Van Gysel and Maarten de Rijke. "Pytrec_eval: An Extremely Fast Python Interface to trec_eval". In: *SIGIR*. ACM, 2018.

[12] Nandan Thakur et al. "BEIR: A Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models". In: *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*. 2021. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

[13] Ehsan Kamalloo et al. *Resources for Brewing BEIR: Reproducible Reference Models and an Official Leaderboard*. 2023. arXiv: 2306.07471 [cs.IR].

[14] Tri Nguyen et al. "MS MARCO: A Human Generated MAchine Reading COmprehension Dataset". In: *arXiv preprint arXiv:1611.09268* (2016).