Information Retrieval Assignment Report

Assignment 3: Text Ranking with Transformer-Based Models

Naveeta Maheshwari (Roll No: 2024AIZ8309)
Sagar Singh (Roll No: 2025SIY7574)

Course: Information Retrieval and Web Search (COL764/COL7364)
Semester: 2025–26
November 28, 2025

1 Introduction

The retrieve-then-rerank paradigm is a standard approach to efficient and accurate ranking. A lexical retriever (BM25) returns a set of candidates quickly; a heavy neural rerank (Hugging Face cross-encoder) computes relevance scores over $\langle q, d \rangle$ pairs and reorders the candidates. The final reported list is truncated to the top-10 for evaluation.

2 Task 1: Implement the "Retrieve and Rerank" framework using BERT

2.1 Implementation Details

2.1.1 Index and Retriever

We used **Pyserini**'s prebuilt `msmarco-passage` Lucene index for BM25 retrieval. For each query, we retrieve top-k documents with $k \in \{10, 25, 50, 75, 100, 150\}$.

2.1.2 Reranker: MiniLM Cross-Encoder

We use the fine-tuned model `cross-encoder/ms-Marco-MiniLM-L-6-v2` from HuggingFace as the heavy reranker.
MiniLM is a distilled version of BERT trained with knowledge distillation from large transformer models, reducing number of layers (6 vs. 12 in BERT-base) while preserving strong contextual relevance matching. This makes it significantly more efficient for inference in retrieval pipelines. As a **cross-encoder**, the model jointly encodes the query–document pair in a single input sequence:

$$\texttt{[CLS]}\ q\ \texttt{[SEP]}\ d\ \texttt{[SEP]}$$

Unlike bi-encoders that independently embed q and d, the cross-encoder allows **full cross-attention** between every query and document token. This enables the model to learn rich, token-level semantic interactions crucial for fine-grained relevance estimation in MS MARCO.

2.1.3 relevance score for reranking

The final `[CLS]` embedding is passed through a classification layer to produce a real-valued **relevance score**:

$$s(q, d_i) = f_{\text{MiniLM}}(\texttt{[CLS]})$$

We apply this scoring to each of the top-k BM25 candidates independently, sort by score, and then output the final top-10 for evaluation.

Since the model must score each $\langle q, d \rangle$ pair separately, complexity grows linearly as $\mathcal{O}(k)$. This accounts for the approximately linear increase in total latency observed in Figure **??**, while yielding consistent gains in NDCG@10 and MRR@10 as k grows.

2.1.4 Chunking

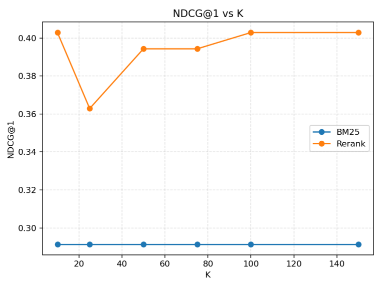
MS MARCO passages are short; in our runs, input lengths remained within the cross-encoder's maximum sequence length. Therefore, no chunking or score aggregation was required.

Outputs and Evaluation. For each k, we produce two TREC-style run files: `bm25_k{K}.run` and `rerank_k{K}.run` with lines of the form: `qid docid rank score`. We compute **NDCG@1/5/10** and **MRR@10** using our evaluation script (qrels format: `qid docid rel`). We also log average per-query latency for BM25, reranking, and total.

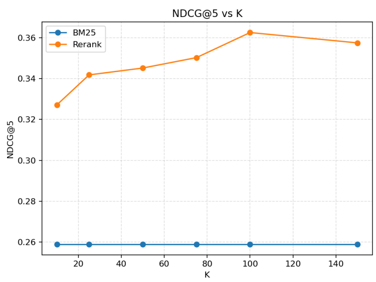
2.2 Results and Observations

2.2.1 NDCG

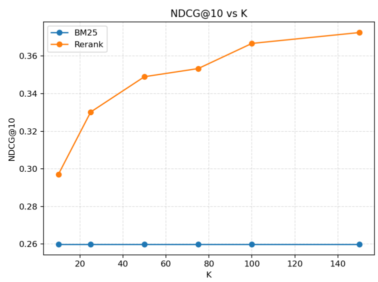
BM25: The BM25 performance remains flat across all k values, as its top-ranked set does not change with a larger retrieval pool.
Reranker: The reranker consistently improves with increasing k, showing higher NDCG scores that stabilize around $k = 100$–150, demonstrating its ability to identify and prioritize truly relevant documents. The corresponding figures are shown in Figure 1.



(a) NDCG@1 vs. k



(b) NDCG@5 vs. k



(c) NDCG@10 vs. k

Figure 1: NDCG scores vs. k for BM25 and Rerank.

2.2.2 MRR@10

BM25: MRR@10 remains nearly constant at 0.534 across all k values.
Reranker: MRR@10 rises from approximately 0.607 at $k = 10$ to around 0.627 at $k = 100$, then fluctuates slightly, reflecting better early precision as relevant documents

are promoted.
The corresponding plot is shown in Fig. 2.

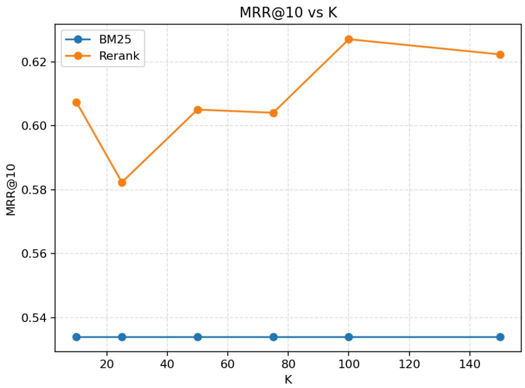


Figure 2: MRR@10 vs. k for BM25 and Rerank.

2.2.3 Best Results for Task1

Table 1: Rerank evaluation results at different k values. Best results per metric are underlined.

k	NDCG@1	NDCG@5	NDCG@10	MRR@10
10	0.403	0.327	0.297	0.607
25	0.363	0.342	0.330	0.582
50	0.394	0.345	0.349	0.605
75	0.394	0.350	0.353	0.604
100	<u>0.403</u>	<u>0.362</u>	0.367	<u>0.627</u>
150	<u>0.403</u>	0.357	<u>0.373</u>	0.622

2.3 Conclusion

The Retrieve & Rerank framework with a fine-tuned BERT cross-encoder consistently outperforms BM25 across all evaluated metrics, with the largest gains observed at higher k values. While reranking effectiveness improves as k increases—reflected in rising NDCG@1/5/10 and MRR@10—marginal returns diminish beyond $k \approx 100$, and latency grows linearly. Therefore, choosing an appropriate k balances improved ranking quality against computational cost, depending on application latency constraints. Overall, our framework demonstrates that augmenting BM25 with cross-encoder reranking delivers significant precision gains with modest additional overhead.

3 Task 2: Comparisons with BERT alternatives

This task investigates two additional reranking baselines on the MS-MARCO dataset:

1. **ELECTRA cross-encoder**
2. **MonoT5 generative reranker**

We evaluate their performance across different candidate pool sizes $k \in \{10, 25, 50, 75, 100, 150\}$ using NDCG@1, NDCG@5, NDCG@10, and MRR@10..

3.1 Model Architecture Comparison

Table 2: Comparison of retrieval and reranking models with differences from BERT.

Model	**Architecture**	**Difference from BERT**	**Key Characteristics**
BM25	Lexical inverted index	Not a neural model; purely term-based	Fast retrieval; no semantic understanding; retrieves candidate pool.
ELECTRA Cross-Encoder (baseline-1)	Transformer encoder fine-tuned for relevance	Uses ELECTRA discriminator pre-training instead of BERT's masked LM; cross-encodes query-document pairs	Strong token interaction modeling; performance improves as k grows; cost $\mathcal{O}(k)$.
MonoT5-Small (baseline-2)	Encoder–decoder (Seq2Seq generative transformer)	Seq2Seq generative model vs. BERT's encoder-only architecture	Weaker discrimination for short passages; performance *degrades* as k grows due to noisy candidates.

3.2 Results and Observation

3.2.1 Results for MonoT5

Table 3: MonoT5 Baseline performance across candidate set size k. Best results per metric are underlined.

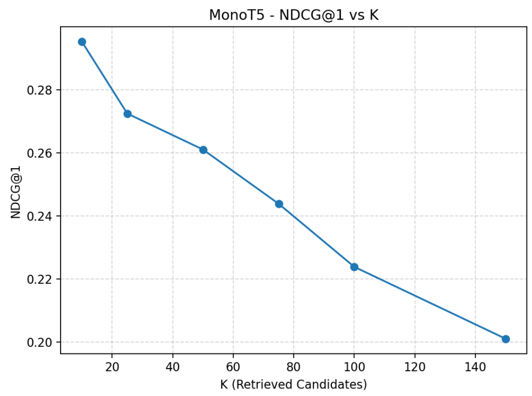
k	NDCG@1	NDCG@5	NDCG@10	MRR@10
10	<u>0.2952</u>	<u>0.2728</u>	<u>0.2635</u>	<u>0.5085</u>
25	0.2724	0.2610	0.2695	0.4837
50	0.2610	0.2315	0.2682	0.4926
75	0.2438	0.2166	0.2569	0.4804
100	0.2238	0.2028	0.2474	0.4701
150	0.2010	0.2064	0.2380	0.4562

3.2.2 Results for ELECTRA Cross-Encoder

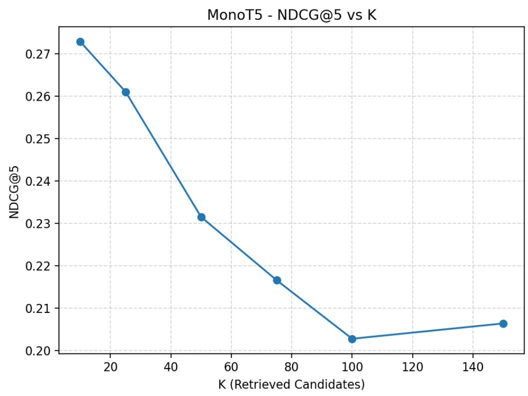
Table 4: ELECTRA Cross-Encoder performance across candidate set size k. Best results per metric are underlined.

k	NDCG@1	NDCG@5	NDCG@10	MRR@10
10	0.3371	0.3179	0.2908	0.5545
25	0.3771	0.3308	0.3165	0.5760
50	0.4314	0.3646	0.3522	0.6379
75	0.4400	0.3773	0.3615	0.6335
100	0.4800	0.3996	0.3856	0.6710
150	0.4429	0.3914	0.3918	0.6653

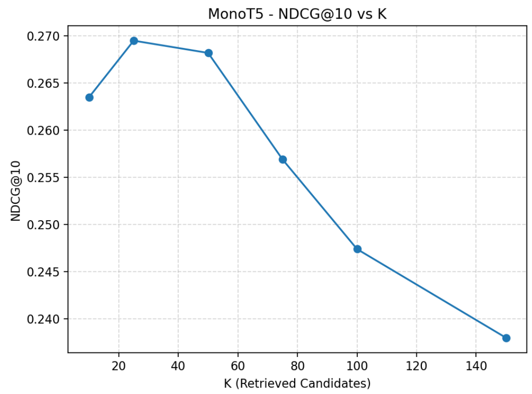
3.3 Plots: MonoT5 Trends



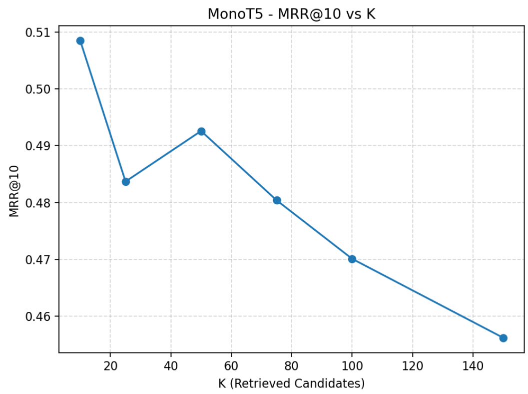
(a) NDCG@1 vs k



(b) NDCG@5 vs k



(c) NDCG@10 vs k

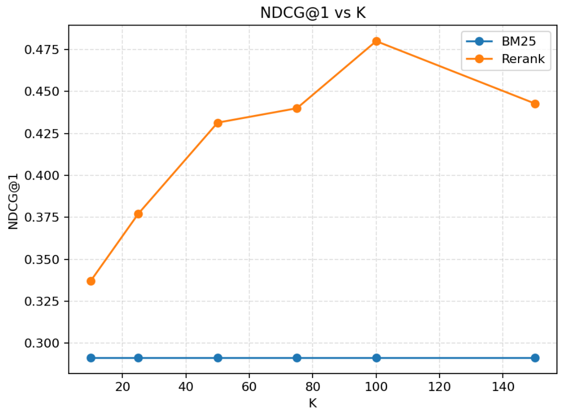


(d) MRR@10 vs k

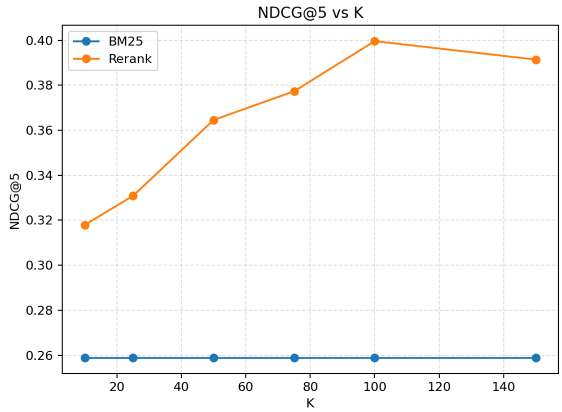
Figure 3: MonoT5 performance metrics across candidate set size k.

3.4 ELECTRA Cross-Encoder: Effectiveness and Latency

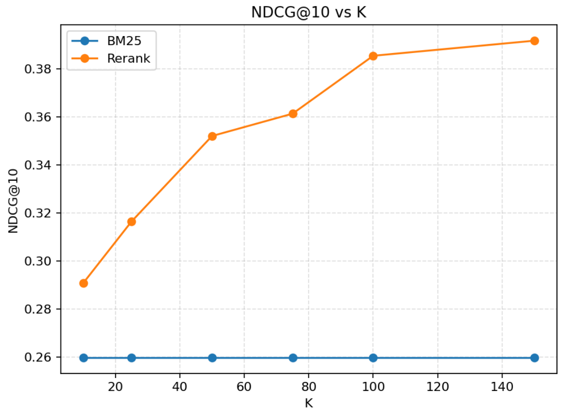
plots for ELECTRA Cross-Encode are give in Fig. 4



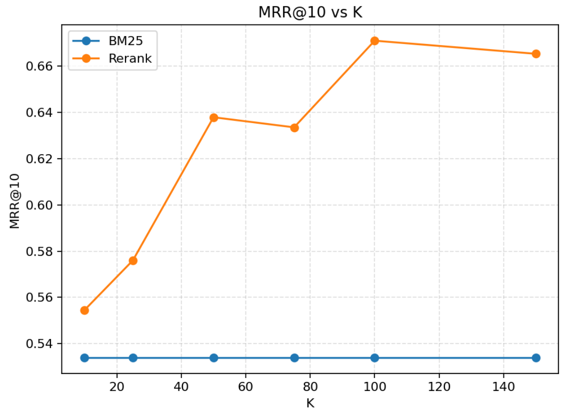
(a) NDCG@1 vs k (ELECTRA)



(b) NDCG@5 vs k (ELECTRA)



(c) NDCG@10 vs k (ELECTRA)



(d) MRR@10 vs k (ELECTRA)

Figure 4: Effectiveness of ELECTRA reranker across candidate pool sizes k.

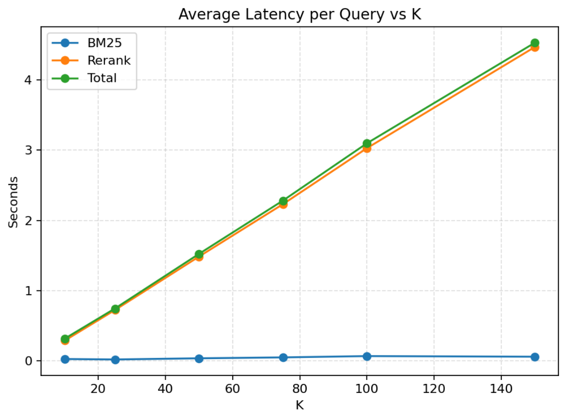


Figure 5: Latency vs k for ELECTRA reranker (BM25, Rerank, Total).

3.5 Analysis & Observations

- **MonoT5 performance drops as k increases:** Expanding the candidate pool introduces more irrelevant passages, which the pointwise T5 classifier struggles to discriminate.

- **ELECTRA cross-encoder improves steadily with k:** Cross-attention allows better exploitation of larger k because relevant items are more likely to enter the pool.
- **BM25-only remains flat:** Final top-10 ranking unchanged across k.
- **Takeaway:** ELECTRA achieves a better accuracy–cost trade-off. MonoT5 is slower and degrades with noise.
  - The ELECTRA Cross-Encoder achieves its best NDCG@1 (0.4800), NDCG@5 (0.3996), and MRR@10 (0.6710) at $k = 100$, while the highest NDCG@10 (0.3918) occurs at $k = 150$.

Conclusion. Task-2 shows that although MonoT5 is a widely-used generative reranker, it is *inferior* to ELECTRA-based cross-encoding for short MS MARCO passages. For production usage, a cross-encoder like MiniLM/ELECTRA appears to be the correct compromise between performance and latency.

4 Task 3: Improving the Best Reranker (Pseudo-Relevance Feedback)

From Tasks 1 and 2, the **ELECTRA cross-encoder** achieved the best ranking performance. In this task, we attempt to further improve its effectiveness by introducing **Pseudo-Relevance Feedback (PRF)** for query expansion prior to reranking.

4.1 Method

For each query:

1. Retrieve top-k documents using BM25.
2. Select the **top-5 documents** as pseudo-relevant.
3. Extract the **top-5 most frequent terms** from those documents.
4. Expand the query by appending these feedback terms.
5. Re-rank the expanded query's candidate set using:

`cross-encoder/ms-marco-electra-base`

This method assumes the highest-ranked BM25 documents are relevant and can provide useful context to enhance the neural reranker's semantic matching ability.

4.2 Results

We evaluate the modified pipeline across $k \in \{10, 25, 50, 75, 100, 150\}$ using the same metrics as in Tasks 1 and 2 (NDCG@1/5/10 and MRR@10).

Finding.

- At $k = 10$, both **MRR@10 and NDCG@1 showed improvement**, indicating that when BM25 returns highly relevant candidates, feedback terms strengthen the query.
- For $k \geq 25$, performance **degraded consistently across all metrics**. The expanded query introduced noise from non-relevant documents, hurting the neural reranker's discrimination ability.

4.3 Analysis

PRF behaves differently in dense reranking environments than in classical IR:

- Neural cross-encoders already capture fine-grained relevance.
- Naively adding frequent lexical terms may:
  - reduce semantic precision,
  - introduce unrelated context,
  - confuse relevance modeling.
- PRF only helps when the initial ranking quality is already high (as seen at $k = 10$).

4.4 Conclusion

While pseudo-relevance feedback is traditionally effective in retrieval, it is **not directly beneficial in neural reranking** unless the feedback set is clean.

Overall, our experiments demonstrate that integrating classical PRF techniques into cross-encoder reranking requires careful control of feedback noise to avoid degrading performance.