Comparative Analysis of Lion and AdamW Optimizers for Cross-Encoder Reranking with MiniLM, GTE, and ModernBERT

Shahil Kumar
IT
IIIT Allahabad
Prayagraj, India
mml2023008@iiita.ac.in

Author 2 Name
Department
Institution
City, Country
email address

***Abstract*—Modern information retrieval systems often employ a two-stage pipeline consisting of an efficient initial retrieval stage followed by a more computationally intensive reranking stage. Cross-encoder models have demonstrated state-of-the-art effectiveness for the reranking task due to their ability to perform deep, contextualized analysis of query-document pairs. The choice of optimizer during the fine-tuning phase can significantly impact the final performance and training efficiency of these models. This paper investigates the impact of using the recently proposed Lion optimizer compared to the widely used AdamW optimizer for fine-tuning cross-encoder rerankers. We fine-tune three distinct transformer models, 'microsoft/MiniLM-L12-H384-uncased', 'Alibaba-NLP/gte-multilingual-base', and 'answerdotai/ModernBERT-base', on the MS MARCO passage ranking dataset using both optimizers. Notably, GTE and ModernBERT support longer context lengths (8192 tokens). The effectiveness of the resulting models is evaluated on the TREC 2019 Deep Learning Track passage ranking task and the MS MARCO development set (for MRR@10). Our experiments, facilitated by the Modal cloud computing platform for GPU resource management, show comparative results across three training epochs. ModernBERT trained with Lion achieved the highest NDCG@10 (0.7225) and MAP (0.5121) on TREC DL 2019, while MiniLM trained with Lion tied with ModernBERT with Lion on MRR@10 (0.5988) on MS MARCO dev. We analyze the performance trends based on standard IR metrics, providing insights into the relative effectiveness of Lion versus AdamW for different model architectures and training configurations in the context of passage reranking.**

***Index Terms*—Information Retrieval, Cross-Encoder, Reranking, Optimizer, Lion Optimizer, AdamW, TREC Deep Learning, MS MARCO, Sentence Transformers, ModernBERT, Long Context, Modal.**

I. Introduction

Information Retrieval (IR) systems aim to satisfy a user's information need, often expressed as a textual query, by returning a ranked list of relevant documents from a large collection. A common and effective architecture for modern search systems is the two-stage retrieve-and-rerank pipeline [1]. The first stage employs computationally efficient methods, such as BM25 [2] or dense vector retrieval [3], to quickly retrieve a candidate set of documents (typically hundreds or thousands) from the entire collection. The second stage then applies a more sophisticated and computationally expensive reranking model to reorder this smaller candidate set, aiming for higher precision at the top ranks.

Cross-encoder models, typically based on transformer architectures like BERT [4], have emerged as state-of-the-art rerankers [5], [6]. Unlike bi-encoder models that encode queries and documents independently, cross-encoders process the query and a candidate document simultaneously (e.g., '[CLS] query [SEP] document [SEP]'). This allows for deep, token-level interaction modeling, leading to superior relevance estimation accuracy, albeit at a higher computational cost suitable only for the second stage. Some models like GTE [7] and ModernBERT [8] support significantly longer input sequences (e.g., 8192 tokens), potentially benefiting the processing of longer passages or documents compared to models with shorter context windows like MiniLM [9].

Fine-tuning these large transformer models effectively is crucial. The choice of optimizer plays a key role in navigating the complex loss landscape and achieving optimal performance. Adam [10] and its variant AdamW [11] are widely used and generally effective optimizers for training deep neural networks. Recently, the Lion (Evo**L**ved Si**g**n M**o**me**n**tum) optimizer [12] was proposed, derived through symbolic mathematics and program search. It claims improved performance and memory efficiency compared to AdamW on various tasks, particularly image classification and vision-language models.

In this work, we investigate the applicability and effectiveness of the Lion optimizer for fine-tuning cross-encoder models specifically for the task of passage reranking in information retrieval. We compare its performance against the standard AdamW optimizer across three different base models with varying characteristics. Our contributions are:

- We fine-tune three distinct transformer-based cross-encoder models ('microsoft/MiniLM-L12-H384-uncased', 'Alibaba-NLP/gte-multilingual-base', and 'answerdotai/ModernBERT-base') on the large-scale MS MARCO passage dataset [13] using both Lion and AdamW optimizers.

- We utilize different training configurations optimized for specific models, including a lower learning rate (2e-6) and a Cosine Annealing scheduler for ModernBERT, contrasting with a higher rate (2e-5) and no scheduler for MiniLM and GTE.
- We evaluate the performance of the fine-tuned models across three training epochs on the TREC 2019 Deep Learning (DL) Track [14] benchmark and the MS MARCO development set [13].
- We provide a comparative analysis of the optimizers' impact on reranking effectiveness using standard IR metrics (NDCG@10, MAP, MRR@10, Recall@10, R-Prec, P@10).
- We utilize the Modal cloud platform [15] for efficient GPU resource management (NVIDIA A100-80GB) and reproducible experimentation.

The rest of the paper is organized as follows: Section II discusses related work. Section III details the models, optimizers, and training approach. Section IV describes the experimental setup, datasets, and evaluation protocol. Section V presents and discusses the results. Finally, Section VI concludes the paper and suggests future work.

II. Related Work

A. Cross-Encoder Reranking

The use of BERT-based models for document reranking was popularized by Nogueira et al. [5], [6]. They demonstrated that fine-tuning BERT as a cross-encoder on relevance labels (like those from MS MARCO) significantly outperforms traditional IR models and even bi-encoder approaches for reranking tasks. Subsequent work explored various transformer architectures [16], training strategies [17], and efficiency improvements [18]. Models like MiniLM [9] offer a balance between effectiveness and efficiency through knowledge distillation. Multilingual models like GTE [7] (General Text Embeddings) provide strong performance across various text tasks, including retrieval, and can be adapted for cross-encoding, supporting longer context lengths. Recently, models like ModernBERT [8] have incorporated architectural improvements such as Rotary Positional Embeddings (RoPE) [19] and Flash Attention [20] to handle even longer sequences (up to 8192 tokens) efficiently and effectively. The 'sentence-transformers' library [21] provides a convenient framework for training and using both bi-encoders and cross-encoders.

B. Optimizers for Deep Learning

Stochastic Gradient Descent (SGD) with momentum remains a fundamental optimizer, but adaptive learning rate methods like AdaGrad [22], RMSprop [23], and Adam [10] often lead to faster convergence in practice for deep learning models. Adam combines momentum with adaptive scaling of learning rates based on estimates of first and second moments of the gradients. AdamW [11] improves upon Adam by decoupling the weight decay regularization from the adaptive learning rate mechanism, often leading to better generalization. The Lion optimizer [12] takes a different approach, using only momentum tracking and a sign operation on the update, resulting in a simpler update rule ('update = sign(momentum) * lr') and potentially requiring less memory due to not storing second moment estimates. Its effectiveness relative to AdamW has been shown primarily in vision and vision-language tasks, motivating its evaluation in the NLP/IR domain.

C. Evaluation Benchmarks

The MS MARCO (Microsoft MAchine Reading COmprehension) dataset [13], [13] has become a standard benchmark for training and evaluating deep learning models for passage retrieval and reranking. It contains real user queries from Bing and human-judged relevant passages. The TREC Deep Learning (DL) Tracks [14], [24] provide challenging test collections built upon MS MARCO, using queries with sparse relevance judgments derived from pooling top results from various participating systems. We use the TREC DL 2019 passage ranking dataset, a standard benchmark for evaluating reranking effectiveness. Evaluation is typically performed using tools like 'trec_eval' [25], which calculates various standard IR metrics. We specifically use the MS MARCO passage dataset's development split for evaluating MRR@10, providing a complementary perspective on a larger query set.

III. Methodology

A. Cross-Encoder Architecture

A cross-encoder model takes a query q and a document d as input, typically concatenating them with special tokens: '[CLS] q [SEP] d [SEP]'. This combined sequence is fed into a transformer model (e.g., MiniLM, GTE, ModernBERT). The output representation corresponding to the '[CLS]' token is then passed through a linear layer followed by a sigmoid activation to predict a relevance score $s(q, d) \in [0, 1]$. During inference for reranking, this score is computed for all candidate documents retrieved in the first stage, and the documents are re-sorted based on these scores. An overview of the architecture is shown in Fig. 1.

B. Base Models

We experiment with three transformer base models from Hugging Face [26]:

- **'microsoft/MiniLM-L12-H384-uncased' [9]:** A distilled version of BERT, designed to be smaller (12 layers, 384 hidden size) and faster while retaining significant performance. It has a standard context length, typically 512 tokens.
- **'Alibaba-NLP/gte-multilingual-base' [7]:** Part of the General Text Embeddings (GTE) family, trained on a large, diverse multilingual corpus. While often used as a bi-encoder, its underlying transformer architecture can be effectively fine-tuned as a cross-encoder. This 'base' version supports a context length of 8192 tokens.
- **'answerdotai/ModernBERT-base' [8]:** A state-of-the-art encoder-only transformer incorporating architectural enhancements like Rotary Positional Embeddings (RoPE) [19] for effective long context handling (up to 8192

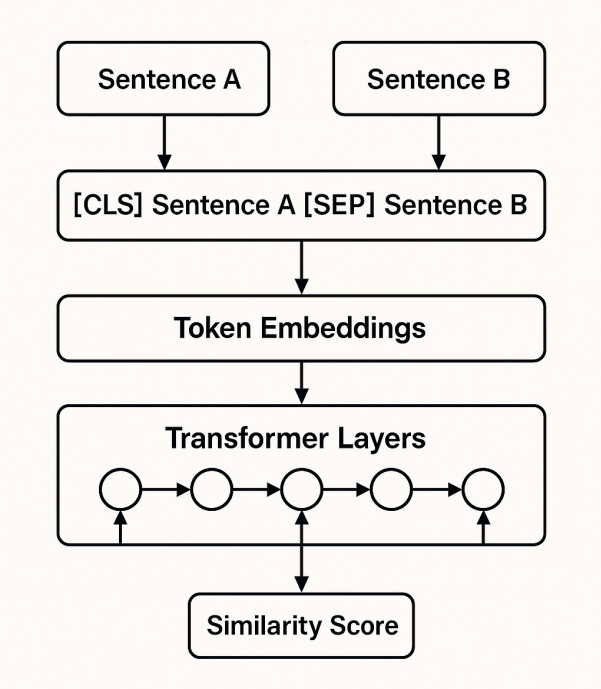


Fig. 1: General Cross-Encoder Architecture for Reranking.

tokens), Flash Attention [20] for speed and memory efficiency, and GeGLU activation functions [27]. It was trained on 2 trillion tokens and demonstrates strong performance on various tasks.

C. Training

We fine-tune the cross-encoders using the MS MARCO passage ranking triplets dataset [13], as processed by 'sentence-transformers' [21]. The original dataset contains tuples of (query, positive passage, negative passage). We convert this into pairs '(query, positive_passage)' with label 1 and '(query, negative_passage)' with label 0. We train on approximately 2 million such pairs. The training objective is to minimize the Binary Cross-Entropy (BCE) loss between the predicted relevance score $\hat{y}$ (output of the sigmoid function) and the true label y (0 or 1). The BCE loss is defined as:

$$\mathcal{L}_{BCE} = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

D. Optimizers

We compare two optimizers using specified parameters:

- **AdamW [11]:** Implemented via `torch.optim.AdamW`. We use a weight decay of 0.01. The learning rate varies by model (see Section IV).
- **Lion [12]:** Implemented via a PyTorch implementation based on the original paper. We use the default betas $\beta_1 = 0.9$, $\beta_2 = 0.99$, and a weight decay of 0.01. The learning rate varies by model. The update rule is as follows:

$$\begin{aligned} c_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ \theta_t &= \theta_{t-1} - \eta \left(\text{sign}(c_t) + \lambda \theta_{t-1}\right) \\ m_t &= \beta_2 m_{t-1} + (1 - \beta_2) g_t \end{aligned}$$

where g_t is the gradient, m_t is the momentum, η is the learning rate, β_1, β_2 are momentum coefficients, and λ is the weight decay.

IV. Experimental Setup

A. Datasets

- **Training:** We use the MS MARCO passage ranking triplets dataset [13], processed into approximately 2 million query-passage pairs with binary labels using the 'sentence-transformers' methodology.
- **Evaluation (Main):** We evaluate on the TREC 2019 Deep Learning Track passage ranking task [14], which contains 43 queries with graded relevance judgments (qrels).
- **Evaluation (MRR@10):** We evaluate Mean Reciprocal Rank at cutoff 10 (MRR@10) on the MS MARCO passage dataset's development split (available from Hugging Face Datasets [13]), containing a larger set of queries with binary relevance.

B. Implementation Details

- **Framework:** We use 'sentence-transformers' [21] ('CrossEncoder') built on PyTorch [28] and Hugging Face Transformers [26].
- **Hyperparameters:** Key training parameters are:
  - Batch Size: 64.
  - Learning Rate:
    - 2e-5 for MiniLM and GTE (both optimizers).
    - 2e-6 for ModernBERT with Lion optimizer.
    - 2e-5 for ModernBERT with adamW optmizer.
  - Scheduler:
    - None for MiniLM and GTE.
    - CosineAnnealingLR Scheduler for ModernBERT, with 'T_max' set to the total number of training steps.
  - Optimizer Params:
    - AdamW: weight_decay=0.01.
    - Lion: betas=(0.9, 0.99), weight_decay=0.01.
  - Warmup Ratio: 0.1 (for LR scheduler if used)
  - Epochs: 3
  - Precision: BF16 enabled.
  - Seed: 12 (for reproducibility).
  - Dataloader Workers: 4.
- **Infrastructure:**
  - Experiments were conducted using the Modal platform [15], leveraging NVIDIA A100-80GB GPUs. Modal managed the environment (CUDA, Python 3.11, libraries) and distributed training.

  - Experiment tracking, visualization of training metrics (such as loss and learning rate, see Fig. 2), and hyperparameter logging were managed using the Weights & Biases (W&B) platform [29].
- **First-Stage Retrieval:** For TREC DL evaluation, we retrieve the top 1000 candidate passages per query using a standard BM25 baseline via Pyserini [30] on the 'msmarco-v1-passage' index. The cross-encoders rerank this candidate set.

C. Evaluation Protocol

- **Reranking:** Each fine-tuned cross-encoder model scores the top-1000 BM25 candidates for each TREC DL 2019 query. Passages are reranked based on these scores.
- **Metrics:** We use 'trec_eval' [25] for TREC DL 2019 evaluation. We report:
  - NDCG@10 (Normalized Discounted Cumulative Gain @10)
  - MAP (Mean Average Precision)
  - Recall@10 (Recall @10)
  - R-Prec (R-Precision)
  - P@10 (Precision @10)

  For MRR@10, we evaluate on the MS MARCO development set.
- **Configurations Tested:** We evaluate each combination of base model and optimizer after each of the 3 training epochs:
  - MiniLM + AdamW (Epochs 1, 2, 3)
  - MiniLM + Lion (Epochs 1, 2, 3)
  - GTE-multilingual-base + AdamW (Epochs 1, 2, 3)
  - GTE-multilingual-base + Lion (Epochs 1, 2, 3)
  - ModernBERT-base + AdamW (Epochs 1, 2, 3)
  - ModernBERT-base + Lion (Epochs 1, 2, 3)

V. Results and Discussion

Table I presents the evaluation results for all configurations on the TREC 2019 Deep Learning Track (NDCG@10, MAP, Recall@10, R-Prec, P@10) and MS MARCO development set (MRR@10) after 1, 2, and 3 epochs of fine-tuning. Best results for each metric are highlighted.

A. Optimizer Performance Comparison

We observe different interactions between the base models and the optimizers:

- **MiniLM:** AdamW generally achieves higher peak performance on TREC DL 2019 metrics (NDCG@10: 0.7127 vs 0.7031; MAP: 0.4908 vs 0.4858), peaking at Epoch 3. Lion achieves the overall best MRR@10 on MS MARCO dev (0.5988 at Epoch 3), but its TREC performance peaks earlier (Epoch 1) and then declines. This suggests AdamW might be more stable over longer training for MiniLM on TREC, while Lion might converge faster or optimize differently for the MS MARCO distribution.
- **GTE:** AdamW clearly outperforms Lion for the GTE model across most metrics and epochs. GTE+AdamW achieves its best TREC performance (NDCG@10=0.7224, MAP=0.5005) early at Epoch 1 or 2, with performance degrading by Epoch 3. GTE+Lion shows weaker performance overall, peaking later (Epoch 2/3) but never reaching the levels of GTE+AdamW.
- **ModernBERT:** Lion significantly outperforms AdamW when training ModernBERT with the specific low learning rate (2e-6) and Cosine Annealing scheduler. ModernBERT+Lion achieves the overall best NDCG@10 (0.7225), MAP (0.5121), R-Prec (0.5183), and P@10 (0.8256) in our experiments. Its performance peaks around Epoch 2 for most TREC metrics. ModernBERT+AdamW, using the same LR and scheduler, performed notably worse, suggesting Lion interacts more favorably with this setup for this model.

These results indicate that the choice of optimizer interacts significantly with the base model architecture and the specific training hyperparameters (learning rate, scheduler). Lion showed particular strength with ModernBERT under its tailored training regime.

B. Model Performance Comparison

Comparing the best results achieved by each model (irrespective of optimizer epoch):

- **ModernBERT (+Lion, E2/E1/E3):** Achieved the highest NDCG@10 (0.7225), MAP (0.5121), R-Prec (0.5183), and P@10 (0.8256) on TREC DL 2019, and tied for the best MRR@10 (0.5988) on MS MARCO dev. This suggests its advanced architecture and long context, combined with the Lion optimizer and specific training strategy, yield superior reranking quality.
- **GTE (+AdamW, E1/E2):** Achieved the next best performance, with NDCG@10 (0.7224) nearly matching ModernBERT+Lion, and achieving the best Recall@10 (0.1733). Its peak performance was reached earlier than ModernBERT's in terms of epochs.
- **MiniLM (+AdamW E3 / +Lion E3):** While performing competitively, MiniLM generally lagged behind GTE and ModernBERT on TREC metrics. However, MiniLM+Lion achieved the joint-highest MRR@10 (0.5988) on the MS MARCO dev set, indicating strong performance on that specific benchmark despite lower TREC scores compared to the larger models.

The results suggest that the larger models with longer context capabilities (GTE, ModernBERT) generally outperform MiniLM on the TREC DL 2019 task. ModernBERT, with its specific optimizations and training configuration using Lion, achieved the overall best effectiveness.

C. Performance Trends Over Epochs and Training Dynamics

Observing Table I, performance is not always monotonic with training epochs.

- GTE+AdamW peaks early (Epoch 1 or 2) and then degrades.
- MiniLM+AdamW shows more gradual improvement or stability up to Epoch 3.

TABLE I: Evaluation Results on TREC-DL 2019 and MS-MARCO Dev Passage Ranking

Base Model	Optimizer	Epoch	NDCG@10	MAP	MRR@10	Recall@10	R-Prec	P@10
MiniLM-L12-H384	AdamW	1	0.7008	0.4814	0.5828	0.1712	0.4899	0.8047
		2	0.7094	0.4891	0.5818	0.1715	0.5017	0.8093
		3	0.7127	0.4908	0.5826	0.1706	0.4962	0.8023
MiniLM-L12-H384	Lion	1	0.7031	0.4858	0.5890	0.1698	0.4904	0.8070
		2	0.6916	0.4755	0.5942	0.1724	0.5041	0.8116
		3	0.6808	0.4706	**0.5988**	0.1701	0.4923	0.8023
GTE-multilingual-base	AdamW	1	0.7224	0.5005	0.5940	0.1733	0.4957	0.8140
		2	0.7203	0.4999	0.5942	**0.1733**	0.5067	0.8163
		3	0.6902	0.4899	0.5972	0.1730	0.5069	0.8140
GTE-multilingual-base	Lion	1	0.6785	0.4754	0.5854	0.1684	0.4849	0.7953
		2	0.6909	0.4921	0.5957	0.1721	0.5053	0.8140
		3	0.6904	0.4912	0.5931	0.1719	0.5041	0.8093
Modern-BERT-base	AdamW	1	0.7105	0.5066	0.5865	0.1678	0.5161	0.8163
		2	0.6839	0.4893	0.5885	0.1634	0.4946	0.7814
		3	0.6959	0.4971	0.5916	0.1623	0.5116	0.7860
Modern-BERT-base	Lion	1	0.7142	**0.5121**	0.5834	0.1689	0.5148	0.8163
		2	**0.7225**	0.5115	0.5907	0.1732	**0.5183**	0.8209
		3	0.7051	0.5020	**0.5988***	0.1722	0.5102	**0.8256**

* MRR@10 is calculated on the MS MARCO v1.1 passage dataset development split. All other metrics are calculated on TREC DL 2019. Best result for each metric highlighted. (*Note: MiniLM+Lion also achieved 0.5988 MRR@10 at epoch 3, (see section A for complete metrics).)

- MiniLM+Lion peaks early on TREC (Epoch 1) but late on MS MARCO MRR (Epoch 3).
- ModernBERT+Lion mostly peaks around Epoch 2.
- ModernBERT+AdamW performance is less consistent, potentially indicating suboptimal interaction with the LR/scheduler setup.

This highlights the importance of evaluating at multiple checkpoints. Additionally, analysis of training/evaluation loss curves, learning rate schedules, and gradient norms (Fig. 2) for ModernBERT provides further insight into the training dynamics, showing convergence patterns under the Cosine Annealing schedule with both optimizers.

D. Comparison with State-of-the-Art

Table II situates our best result (ModernBERT + Lion, Epoch 2) relative to other published results on TREC DL 2019 passage ranking.

Our best model, ModernBERT+Lion (NDCG@10 = 0.7225), achieves the highest performance among the models listed in Table II, surpassing strong baselines such as BM25 (0.506), dense retrieval models like ANCE (0.648) and BM25 Neg (0.664), hybrid models such as Bi-encoder (PoolAvg) + doc2query-T5 (0.719), and cross-encoder models including cross-encoder/ms-marco-electra-base (0.7199). Our GTE+AdamW model achieves nearly identical performance (0.7224), while our MiniLM+AdamW variant scores 0.7127, remaining competitive. These results highlight the effectiveness of our optimizer choices and training strategies for base-sized models.

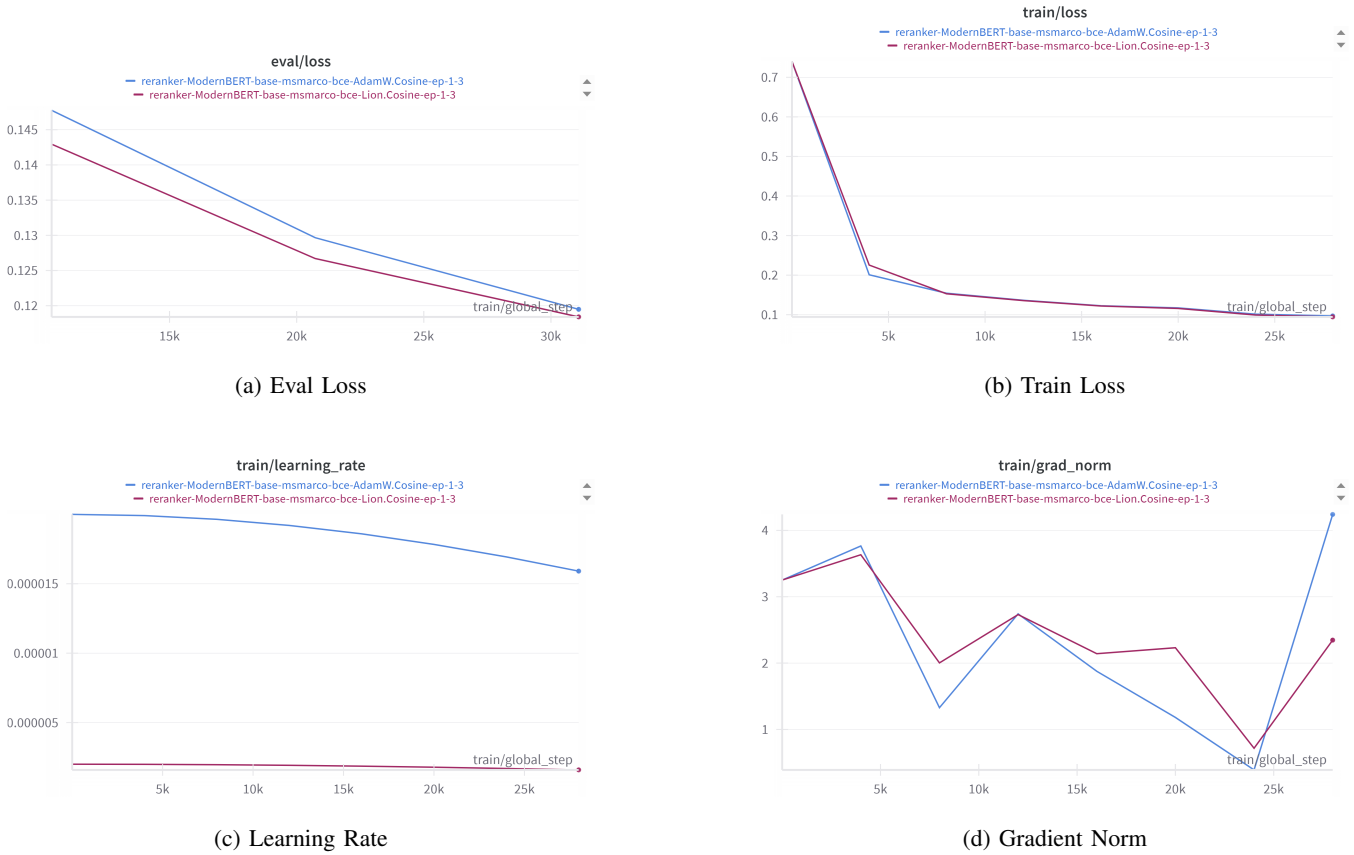
VI. Conclusion

In this paper, we presented a comparative study of the Lion and AdamW optimizers for fine-tuning three

TABLE II: Comparison with Other Systems on TREC DL 2019 (NDCG@10)

Model/System	NDCG@10	Reference
Sparse Retrieval (single stage)		
BM25	0.506	[31]
DeepCT	0.551	[17]
doc2query-T5	0.642	[16]
Dense Retrieval Models		
ANCE	0.648	[32]
Rand Neg	0.605	[32]
NCE Neg	0.602	[32]
BM25 Neg	0.664	[32]
DPR (BM25 + Rand Neg)	0.653	[32]
BM25 $\rightarrow$ Rand	0.609	[32]
BM25 $\rightarrow$ NCE Neg	0.608	[32]
BM25 $\rightarrow$ BM25 + Rand	0.648	[32]
Hybrid dense + sparse (single stage)		
Bi-encoder (PoolAvg) + BM25	0.701	[31]
Bi-encoder (TCT-ColBERT) + BM25	0.714	[31]
Bi-encoder (PoolAvg) + doc2query-T5	0.719	[31]
Cross-Encoder/Reranking Models		
BERT-base Cross-Encoder	0.634	[16]
BERT-large Cross-Encoder	0.654	[16]
pt-bert-base-uncased-msmarco	0.709	[33]
cross-encoder/ms-marco-electra-base	0.719	[21]
cross-encoder/ms-marco-TinyBERT-L-2-v2	0.698	[21]
Our Best (ModernBERT+Lion E2)	**0.7225**	This work
Our GTE+AdamW E1/E2	**0.7224** / 0.7203	This work
Our MiniLM+AdamW E3	0.7127	This work

NDCG@10 scores are reported as evaluated on the TREC DL 2019 dataset using standard settings from cited references. Minor variations may occur due to differences in implementation, random seeds, or reranking candidate sets.



(a) Eval Loss

(b) Train Loss

(c) Learning Rate

(d) Gradient Norm

Fig. 2: Training Dynamics for ModernBERT-base with Lion and AdamW optimizers. Each plot shows a different training metric over training steps.

different transformer models (‘microsoft/MiniLM-L12-H384-uncased‘, ‘Alibaba-NLP/gte-multilingual-base‘, ‘answerdotai/ModernBERT-base‘) as cross-encoders for passage reranking. Training was performed on MS MARCO, and evaluation used TREC 2019 Deep Learning Track and MS MARCO dev benchmarks. Our findings indicate:

- The choice of optimizer significantly interacts with the base model and training hyperparameters (LR, scheduler).
- For ModernBERT, using a low learning rate (2e-6) and Cosine Annealing scheduler, Lion substantially outperformed AdamW, achieving the best overall TREC DL 2019 results (NDCG@10 = 0.7225, MAP = 0.5121 at Epochs 1-2).
- For GTE, AdamW (with LR 2e-5, no scheduler) was more effective than Lion, achieving strong results (NDCG@10 = 0.7224) comparable to ModernBERT+Lion.
- For MiniLM, AdamW yielded slightly better TREC performance over 3 epochs, while Lion achieved the best MS MARCO dev MRR@10 (tied with ModernBERT+Lion).
- Models with longer context capabilities (GTE, ModernBERT) generally outperformed MiniLM on the TREC DL task.
- Performance varied across epochs, emphasizing the need for checkpoint selection based on validation performance.

Our results show that Lion can be a highly effective optimizer for cross-encoder training, particularly for newer architectures like ModernBERT when paired with appropriate learning rate strategies. However, AdamW remains a strong baseline, especially for models like GTE in this setup. The Modal platform proved effective for managing the required GPU resources and experimental setup.

Future work could involve a more thorough hyperparameter search for both optimizers across all models, particularly exploring different learning rates and schedules for Lion with MiniLM and GTE. Investigating the impact of the 8K context length more directly (e.g., on datasets with longer documents) and analyzing memory usage differences between optimizers would also be valuable.

ACKNOWLEDGMENT

The authors acknowledge Modal Labs (https://modal.com/) for providing the cloud computing platform and GPU(s) resources (NVIDIA A100-80GB) used for conducting the experiments presented in this paper.

REFERENCES

[1] M. Hu, Y. Peng, Z. Huang, and D. Li, "Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension," 2019. [Online]. Available: https://arxiv.org/abs/1906.04618

[2] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[3] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih, "Dense passage retrieval for open-domain question answering," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6769–6781.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[5] R. Nogueira and K. Cho, "Passage re-ranking with bert," 2020. [Online]. Available: https://arxiv.org/abs/1901.04085

[6] R. Nogueira, Z. Yang, K. Cho, and J. Lin, "Document ranking with a pretrained sequence-to-sequence model," in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1124–1136. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.102

[7] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, "Towards general text embeddings with multi-stage contrastive learning," 2023. [Online]. Available: https://arxiv.org/abs/2308.03281

[8] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, N. Cooper, G. Adams, J. Howard, and I. Poli, "Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference," 2024. [Online]. Available: https://arxiv.org/abs/2412.13663

[9] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," 2020. [Online]. Available: https://arxiv.org/abs/2002.10957

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[11] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019. [Online]. Available: https://arxiv.org/abs/1711.05101

[12] X. Chen, C. Liang, D. Huang, E. Real, K. Wang, Y. Liu, H. Pham, X. Dong, T. Luong, C.-J. Hsieh, Y. Lu, and Q. V. Le, "Symbolic discovery of optimization algorithms," 2023. [Online]. Available: https://arxiv.org/abs/2302.06675

[13] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, "MS MARCO: A human generated machine reading comprehension dataset," *CoRR*, vol. abs/1611.09268, 2016. [Online]. Available: http://arxiv.org/abs/1611.09268

[14] N. Craswell, B. Mitra, E. Yilmaz, D. Campos, and E. M. Voorhees, "Overview of the trec 2019 deep learning track," 2020. [Online]. Available: https://arxiv.org/abs/2003.07820

[15] Modal, "Modal: Serverless compute for ai and data workflows," https://modal.com/, 2025, accessed: 20-04-2025.

[16] J. Lin, R. Nogueira, and A. Yates, "Pretrained transformers for text ranking: Bert and beyond," in *Synthesis Lectures on Human Language Technologies*, 2021. [Online]. Available: https://arxiv.org/abs/2010.06467

[17] L. Gao, Z. Dai, T. Chen, Z. Fan, B. V. Durme, and J. Callan, "Complementing lexical retrieval with semantic residual embedding," 2021. [Online]. Available: https://arxiv.org/abs/2004.13969

[18] S. Hofstätter, S. Althammer, M. Schröder, M. Sertkan, and A. Hanbury, "Improving efficient neural ranking models with cross-architecture knowledge distillation," 2021. [Online]. Available: https://arxiv.org/abs/2010.02666

[19] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "Roformer: Enhanced transformer with rotary position embedding," 2023. [Online]. Available: https://arxiv.org/abs/2104.09864

[20] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "Flashattention: Fast and memory-efficient exact attention with io-awareness," 2022. [Online]. Available: https://arxiv.org/abs/2205.14135

[21] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 11 2019, pp. 3982–3992. [Online]. Available: https://arxiv.org/abs/1908.10084

[22] J. C. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, Feb. 2011. [Online]. Available: https://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

[23] T. Tieleman and G. Hinton, "Lecture 6.5—rmsprop: Divide the gradient by a running average of its recent magnitude," Coursera: Neural Networks for Machine Learning, 2012, accessed: Apr. 22, 2025. [Online]. Available: https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

[24] N. Craswell, B. Mitra, E. Yilmaz, and D. Campos, "Overview of the trec 2020 deep learning track," 2021. [Online]. Available: https://arxiv.org/abs/2102.07662

[25] National Institute of Standards and Technology (NIST), "trec_eval information retrieval evaluation software," https://github.com/usnistgov/trec_eval, 2024.

[26] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[27] N. Shazeer, "Glu variants improve transformer," 2020. [Online]. Available: https://arxiv.org/abs/2002.05202

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," 2019. [Online]. Available: https://arxiv.org/abs/1912.01703

[29] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[30] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, "Pyserini: A Python toolkit for reproducible information retrieval research with sparse and dense representations," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2356–2360. [Online]. Available: https://dl.acm.org/doi/10.1145/3404835.3463238

[31] S.-C. Lin, J.-H. Yang, and J. Lin, "Distilling dense representations for ranking using tightly-coupled teachers," 2020. [Online]. Available: https://arxiv.org/abs/2010.11386

[32] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," 2020. [Online]. Available: https://arxiv.org/abs/2007.00808

[33] C. Thienes and J. Pertschuk, "Nboost: Neural boosting search results," https://github.com/koursaros-ai/nboost, 2019.

APPENDIX

This appendix presents comprehensive evaluation metrics for all model configurations tested in our experiments. While the main paper focuses on key performance indicators, these detailed tables provide additional insights into model behavior across various retrieval effectiveness measures.

A. Primary Ranking Metrics

Table III presents the main effectiveness metrics including MAP, NDCG@10, MRR@10, and Precision@10 for all tested configurations.

B. Precision and Recall Analysis

Tables IV and V provide detailed precision and recall values at different cutoff thresholds, revealing the models' behavior at various result list depths.

C. NDCG and MAP Metrics

Tables VI and VII provide a more granular view of ranking quality through normalized discounted cumulative gain and mean average precision at multiple cutoff points.

D. Advanced Retrieval Metrics

Table VIII presents R-Precision and interpolated precision metrics that provide insights into precision at different recall levels.

E. User-Focused Metrics

Table IX presents metrics that focus on user satisfaction, including success rates and rank-biased precision.

F. Metric Definitions

Table X provides formal definitions of all evaluation metrics used in our experimental analysis.

TABLE III: Primary ranking metrics comparing cross-encoder models across training epochs.

Model	Optimizer	Epoch	MAP	NDCG@10	MRR@10	P@10	Recip_Rank	R-Prec	bpref
GTE	AdamW	1	0.5005	0.7224	0.5940	0.8116	**0.9814**	0.4964	0.6068
		2	0.4999	0.7203	0.5942	**0.8256**	0.9523	0.5018	0.6074
		3	0.4899	0.6902	0.5972	0.8093	0.9279	0.5017	0.6075
	Lion	1	0.4794	0.6970	0.5854	0.7884	0.9612	0.4814	0.6079
		2	0.4577	0.6792	0.5957	0.7721	0.9477	0.4642	0.6013
		3	0.4521	0.6571	0.5931	0.7488	0.9399	0.4662	0.5972
MiniLM	AdamW	1	0.4814	0.7008	0.5828	0.8000	0.9465	0.4884	0.6016
		2	0.4891	0.7094	0.5818	0.8116	**0.9814**	0.4916	0.6030
		3	0.4908	0.7127	0.5826	0.8116	0.9806	0.4943	0.6048
	Lion	1	0.4858	0.7031	0.5890	0.8140	0.9457	0.4952	0.6010
		2	0.4755	0.6916	0.5942	0.8070	0.9496	0.4803	0.5992
		3	0.4706	0.6808	**0.5988**	0.8070	0.9419	0.4809	0.5994
ModernBERT	AdamW	1	0.5066	0.7105	0.5866	0.8163	0.9574	0.5161	0.6139
		2	0.4893	0.6839	0.5886	0.7814	0.9341	0.4946	0.6152
		3	0.4971	0.6959	0.5916	0.7860	0.9523	0.5116	0.6128
	Lion	1	**0.5121**	0.7142	0.5834	0.8163	0.9593	0.5148	0.6140
		2	0.5115	**0.7225**	0.5908	0.8209	0.9767	**0.5183**	**0.6156**
		3	0.5020	0.7051	**0.5988**	**0.8256**	0.9322	0.5102	0.6151

TABLE IV: Precision metrics at different cutoff levels.

Model	Optimizer	Epoch	P@5	P@10	P@15	P@20	P@30	P@100	P@200	P@500	P@1000
GTE	AdamW	1	0.8930	0.8116	0.7783	0.7372	0.6419	0.3965	0.2664	0.1238	0.0654
		2	0.8698	0.8256	0.7752	0.7326	0.6395	0.3963	0.2641	0.1230	0.0654
		3	0.8558	0.8093	0.7550	0.7128	0.6349	0.3942	0.2634	0.1225	0.0654
	Lion	1	0.8558	0.7884	0.7364	0.7070	0.6318	0.3893	0.2591	0.1228	0.0654
		2	0.8326	0.7721	0.7302	0.6802	0.6054	0.3740	0.2528	0.1227	0.0654
		3	0.8140	0.7488	0.7147	0.6907	0.6140	0.3740	0.2543	0.1231	0.0654
MiniLM	AdamW	1	0.8698	0.8000	0.7628	0.7151	0.6450	0.3870	0.2564	0.1242	0.0654
		2	0.8698	0.8116	0.7674	0.7186	0.6488	0.3895	0.2580	0.1247	0.0654
		3	0.8558	0.8116	0.7705	0.7256	0.6442	0.3891	0.2580	0.1245	0.0654
	Lion	1	0.8651	0.8140	0.7783	0.7233	0.6434	0.3847	0.2583	0.1242	0.0654
		2	0.8744	0.8070	0.7643	0.7081	0.6264	0.3821	0.2530	0.1240	0.0654
		3	0.8465	0.8070	0.7442	0.6930	0.6194	0.3809	0.2547	0.1240	0.0654
ModernBERT	AdamW	1	0.8651	0.8163	0.7767	0.7349	0.6612	0.3993	0.2679	0.1262	0.0654
		2	0.8605	0.7814	0.7333	0.6953	0.6333	0.3912	0.2658	0.1257	0.0654
		3	0.8465	0.7860	0.7411	0.6988	0.6450	0.3958	0.2655	0.1253	0.0654
	Lion	1	0.8791	0.8163	0.7783	0.7314	0.6574	0.4033	0.2691	0.1265	0.0654
		2	0.8698	**0.8209**	0.7674	0.7279	0.6574	**0.4016**	0.2678	0.1267	0.0654
		3	0.8651	0.8256	0.7504	0.7093	0.6488	0.3967	0.2651	0.1263	0.0654

TABLE V: Recall metrics at different cutoff levels.

Model	Optimizer	Epoch	Recall@5	Recall@10	Recall@15	Recall@20	Recall@30	Recall@100	Recall@200	Recall@500	Recall@1000
GTE	AdamW	1	0.1051	0.1689	0.2286	0.2815	0.3376	0.5501	0.6573	0.7161	0.7389
		2	0.1041	0.1760	0.2295	0.2753	0.3366	0.5538	0.6511	0.7130	0.7389
		3	0.1051	0.1715	0.2256	0.2692	0.3376	0.5489	0.6501	0.7102	0.7389
	Lion	1	0.1002	0.1614	0.2158	0.2684	0.3369	0.5401	0.6384	0.7101	0.7389
		2	0.0973	0.1610	0.2151	0.2545	0.3153	0.5211	0.6273	0.7105	0.7389
		3	0.0929	0.1542	0.2070	0.2574	0.3221	0.5181	0.6227	0.7124	0.7389
MiniLM	AdamW	1	0.1027	0.1652	0.2225	0.2645	0.3362	0.5426	0.6396	0.7187	0.7389
		2	0.1013	0.1668	0.2239	0.2676	0.3401	0.5446	0.6417	0.7223	0.7389
		3	0.1007	0.1702	0.2257	0.2719	0.3386	0.5427	0.6396	0.7189	0.7389
	Lion	1	0.1022	0.1688	0.2257	0.2704	0.3400	0.5338	0.6413	0.7154	0.7389
		2	0.1025	0.1641	0.2200	0.2613	0.3289	0.5338	0.6268	0.7145	0.7389
		3	0.1001	0.1676	0.2156	0.2542	0.3253	0.5249	0.6341	0.7143	0.7389
ModernBERT	AdamW	1	0.0995	0.1678	0.2245	0.2731	0.3475	0.5520	0.6539	0.7224	0.7389
		2	0.0979	0.1634	0.2148	0.2603	0.3321	0.5443	0.6520	0.7198	0.7389
		3	0.1019	0.1623	0.2184	0.2627	0.3380	0.5476	0.6461	0.7198	0.7389
	Lion	1	0.1052	0.1689	0.2276	0.2733	0.3431	0.5542	0.6553	0.7233	0.7389
		2	0.1048	**0.1732**	0.2245	0.2699	0.3469	**0.5608**	**0.6553**	**0.7233**	0.7389
		3	0.1037	0.1722	0.2185	0.2640	0.3425	0.5570	0.6525	0.7194	0.7389

TABLE VI: NDCG metrics at different cutoff levels.

Model	Optimizer	Epoch	NDCG	NDCG@5	NDCG@10	NDCG@20	NDCG@30	NDCG@100	NDCG@500	NDCG@1000
GTE	AdamW	1	0.6987	0.7567	0.7224	0.7071	0.6802	0.6591	0.6897	0.6987
		2	0.6949	0.7343	0.7203	0.6967	0.6687	0.6550	0.6845	0.6949
		3	0.6848	0.7080	0.6902	0.6703	0.6558	0.6426	0.6733	0.6848
	Lion	1	0.6865	0.7229	0.6970	0.6786	0.6638	0.6433	0.6750	0.6865
		2	0.6772	0.7008	0.6792	0.6580	0.6405	0.6248	0.6661	0.6772
		3	0.6674	0.6801	0.6571	0.6450	0.6310	0.6124	0.6573	0.6674
MiniLM	AdamW	1	0.6857	0.7207	0.7008	0.6833	0.6698	0.6402	0.6778	0.6857
		2	0.6910	0.7345	0.7094	0.6908	0.6779	0.6471	0.6843	0.6910
		3	0.6910	0.7244	0.7127	0.6949	0.6749	0.6474	0.6834	0.6910
	Lion	1	0.6841	0.7163	0.7031	0.6829	0.6667	0.6369	0.6752	0.6841
		2	0.6790	0.7099	0.6916	0.6706	0.6541	0.6295	0.6692	0.6790
		3	0.6770	0.6837	0.6808	0.6584	0.6479	0.6268	0.6667	0.6770
ModernBERT	AdamW	1	0.6937	0.7171	0.7105	0.6936	0.6831	0.6555	0.6877	0.6937
		2	0.6869	0.7112	0.6839	0.6701	0.6589	0.6431	0.6795	0.6869
		3	0.6910	0.7156	0.6959	0.6729	0.6699	0.6497	0.6832	0.6910
	Lion	1	0.6984	0.7285	0.7142	0.6991	0.6825	0.6610	0.6925	0.6984
		2	**0.7018**	**0.7381**	**0.7225**	**0.6978**	**0.6852**	**0.6638**	**0.6955**	0.7018
		3	0.6930	0.7165	0.7051	0.6797	0.6722	0.6542	0.6857	0.6930

TABLE VII: MAP metrics at different cutoff levels and related metrics.

Model	Optimizer	Epoch	MAP	MAP@10	MAP@20	MAP@30	MAP@100	MAP@200	MAP@500	MAP@1000	infAP	gm_map
GTE	AdamW	1	0.5005	0.1601	0.2499	0.2915	0.4226	0.4765	0.4964	0.5005	0.5005	0.4143
		2	0.4999	0.1632	0.2463	0.2904	0.4247	0.4756	0.4957	0.4999	0.4999	0.4109
		3	0.4899	0.1555	0.2350	0.2827	0.4136	0.4654	0.4855	0.4899	0.4899	0.3995
	Lion	1	0.4794	0.1468	0.2274	0.2771	0.4039	0.4538	0.4755	0.4794	0.4794	0.3878
		2	0.4577	0.1464	0.2201	0.2631	0.3810	0.4294	0.4534	0.4577	0.4577	0.3623
		3	0.4521	0.1378	0.2145	0.2589	0.3747	0.4240	0.4477	0.4521	0.4521	0.3525
MiniLM	AdamW	1	0.4814	0.1479	0.2272	0.2809	0.4062	0.4533	0.4775	0.4814	0.4814	0.3920
		2	0.4891	0.1531	0.2329	0.2882	0.4148	0.4618	0.4854	0.4891	0.4891	0.3975
		3	0.4908	0.1553	0.2383	0.2895	0.4159	0.463	0.4871	0.4908	0.4908	0.3972
	Lion	1	0.4858	0.1534	0.2344	0.2845	0.4087	0.4584	0.4821	0.4858	0.4858	0.3919
		2	0.4755	0.1492	0.2254	0.2735	0.4006	0.4459	0.4717	0.4755	0.4755	0.3823
		3	0.4706	0.1507	0.2207	0.2690	0.3940	0.4418	0.467	0.4706	0.4706	0.3736
ModernBERT	AdamW	1	0.5066	0.1535	0.2389	0.2942	0.4274	0.482	0.5038	0.5066	0.5066	0.4045
		2	0.4893	0.1493	0.2280	0.2803	0.4097	0.4647	0.4865	0.4893	0.4893	0.3875
		3	0.4971	0.1504	0.2304	0.2848	0.4192	0.4717	0.4941	0.4971	0.4971	0.3988
	Lion	1	0.5121	0.1577	0.2448	0.2979	0.4347	0.4883	0.5094	0.5121	0.5121	0.4162
		2	**0.5115**	**0.1616**	**0.2417**	**0.2973**	**0.4333**	0.4866	0.509	0.5115	**0.5115**	**0.4157**
		3	0.5020	0.1583	0.2333	0.2892	0.4239	0.4763	0.4993	0.502	0.5020	0.4071

TABLE VIII: R-Precision and interpolated precision metrics.

Model	Config	R-Prec	R-Prec@0.8	R-Prec@1.0	iprec@0.0	iprec@0.1	iprec@0.3	iprec@0.5	iprec@0.7	iprec@0.9
GTE	AdamW-1	0.4964	0.5617	0.4964	0.9841	0.8741	0.7176	0.5037	0.3295	0.1439
	AdamW-2	0.5018	0.5644	0.5018	0.9730	0.8796	0.7093	0.5027	0.3339	0.1399
	AdamW-3	0.5017	0.5616	0.5017	0.9611	0.8614	0.6942	0.4980	0.3404	0.1395
	Lion-1	0.4814	0.5419	0.4814	0.9667	0.8426	0.6601	0.4907	0.3576	0.1233
	Lion-2	0.4642	0.5238	0.4642	0.9682	0.8377	0.6283	0.4549	0.3144	0.1182
	Lion-3	0.4662	0.5236	0.4662	0.9519	0.8008	0.6289	0.4653	0.3158	0.1218
MiniLM	AdamW-1	0.4884	0.5502	0.4884	0.9637	0.8546	0.6592	0.4804	0.3474	0.1258
	AdamW-2	0.4916	0.5523	0.4916	0.9821	0.8584	0.6614	0.4856	0.3537	0.1251
	AdamW-3	0.4943	0.5611	0.4943	0.9817	0.8641	0.6724	0.4877	0.3469	0.1262
	Lion-1	0.4952	0.5495	0.4952	0.9727	0.8613	0.6698	0.4836	0.3387	0.1326
	Lion-2	0.4803	0.5335	0.4803	0.9728	0.8450	0.6441	0.4545	0.3422	0.1261
	Lion-3	0.4809	0.5240	0.4809	0.9625	0.8511	0.6443	0.4543	0.3370	0.1278
ModernBERT	AdamW-1	0.5161	0.5804	0.5161	0.9688	0.8676	0.6739	0.5126	0.3858	0.1644
	AdamW-2	0.4946	0.5492	0.4946	0.9583	0.8429	0.6547	0.4956	0.3741	0.1464
	AdamW-3	0.5116	0.5638	0.5116	0.9614	0.8460	0.6781	0.5001	0.3774	0.1489
	Lion-1	0.5148	0.5848	0.5148	0.9734	0.8767	0.6892	0.5212	0.3766	0.1566
	Lion-2	**0.5183**	**0.5830**	**0.5183**	**0.9901**	**0.8804**	**0.7052**	**0.5121**	0.3853	**0.1630**
	Lion-3	0.5102	0.5629	0.5102	0.9660	0.8487	0.6954	0.5021	**0.3728**	0.1597

TABLE IX: Success and utility metrics.

Model	Config	Success@1	Success@5	Success@10	Utility	RBP
GTE	AdamW-1	0.9767	1.0000	1.0000	-869.1163	0.5638
	AdamW-2	0.9302	1.0000	1.0000	-869.1163	0.5576
	AdamW-3	0.8837	1.0000	1.0000	-869.1163	0.5418
	Lion-1	0.9302	1.0000	1.0000	-869.1163	0.5428
	Lion-2	0.9070	1.0000	1.0000	-869.1163	0.5279
	Lion-3	0.9070	0.9767	1.0000	-869.1163	0.5156
MiniLM	AdamW-1	0.9070	1.0000	1.0000	-869.1163	0.5484
	AdamW-2	0.9767	1.0000	1.0000	-869.1163	0.5520
	AdamW-3	0.9767	0.9767	1.0000	-869.1163	0.5531
	Lion-1	0.9070	1.0000	1.0000	-869.1163	0.5499
	Lion-2	0.9070	1.0000	1.0000	-869.1163	0.5430
	Lion-3	0.8837	1.0000	1.0000	-869.1163	0.5346
ModernBERT	AdamW-1	0.9302	1.0000	1.0000	-869.1163	0.5586
	AdamW-2	0.8837	1.0000	1.0000	-869.1163	0.5393
	AdamW-3	0.9302	1.0000	1.0000	-869.1163	0.5437
	Lion-1	0.9302	1.0000	1.0000	-869.1163	0.5587
	Lion-2	**0.9535**	1.0000	1.0000	-869.1163	**0.5581**
	Lion-3	0.8837	1.0000	1.0000	-869.1163	0.5475

TABLE X: Definitions of key evaluation metrics used in this paper.

Metric	Definition
MAP	Mean Average Precision. The mean of average precision scores across all queries, where average precision is the average of precision values calculated at every position where a relevant document is retrieved. Range: [0,1].
NDCG@k	Normalized Discounted Cumulative Gain at rank k. Measures ranking quality with graded relevance and position-based discount. Range: [0,1].
P@k	Precision at k. The proportion of retrieved documents in the top k results that are relevant. Range: [0,1].
Recall@k	Recall at k. The proportion of all relevant documents that are retrieved in the top k results. Range: [0,1].
MRR@10	Mean Reciprocal Rank at 10. Average of the reciprocal ranks of the first relevant document within the top 10 results. Range: [0,1].
R-Prec	R-Precision. Precision after R documents retrieved, where R is the number of relevant documents for the query. Range: [0,1].
bpref	Binary Preference. Measures ranking quality when relevance judgments are incomplete by considering known relevant/non-relevant document pairs. Range: [0,1].
Success@k	Indicates whether at least one relevant document is retrieved in the top k results (binary). Range: $\{0,1\}$.
RBP	Rank-Biased Precision. Geometric discount-based measure of user satisfaction that models a user who browses through ranked results with a certain persistence. Range: [0,1].
Recip_Rank	Reciprocal Rank. The reciprocal of the rank at which the first relevant document is retrieved. Range: (0,1].