UTILIZING DEEP LEARNING FOR HYBRID RETRIEVAL AND MULTI-STAGE TEXT RANKING

Keerrtivardhan Goyal - 202103007

Yash Mashru - 202103045

Sanchit Satija - 202103054

INTRODUCTION

The study introduces a multi-stage ranking system that strategically integrates sparse and dense retrieval techniques, aiming to enhance text retrieval and ranking within information retrieval systems.

- Challenge: Deep learning methods demand vast datasets, hindering traditional information retrieval.
- Solution: The Deep Learning Track at TREC addresses this by supplying large-scale datasets for blind evaluation.
- Optimization: Hybrid retrieval methodologies and diverse backbone networks promise a new era of comprehensive ranking performance.

DATASET DESCRIPTION AND EVALUATION

MS MARCO dataset is union of the top ten passage lists for the one million queries, giving 8.8 million distinct passages. The passage corpus is in jsonl format.

Queries originate from real Bing user search logs.

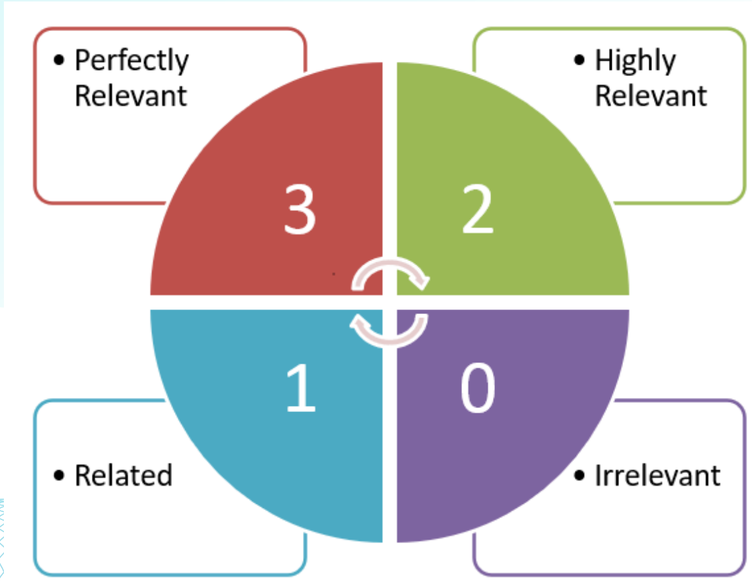
Human annotators judged the relevance of up to 10 passages per query.

Training: 80% of queries, Development: 10%, Evaluation: 10%

DATASET DESCRIPTION AND EVALUATION

Graded Relevance

Normalized Cumulative Discount Gain (nDCG@k)} metric strikes an optimal balance for graded relevance assessments



The passage corpus is in jsonl format. Each passage has

pid

passage

spans

docid

MODEL ARCHITECTURE

Retrieval :-

1.) BM25
2.) Elastic BM25
3.) SPARTA
4.) SBERT(trained with inbatch negatives)
5.) SBERT(pre-trained)
6.) SBERT with faiss indexing(HNSW)
7.) SBERT with faiss indexing(FLAT)

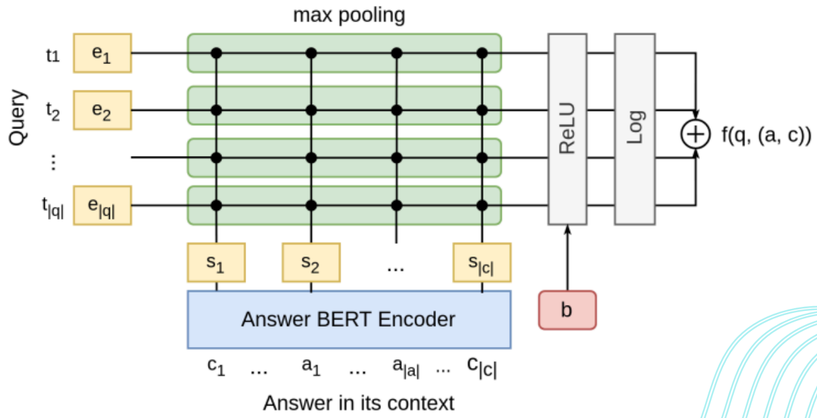
Retrieval + Re-ranking:-

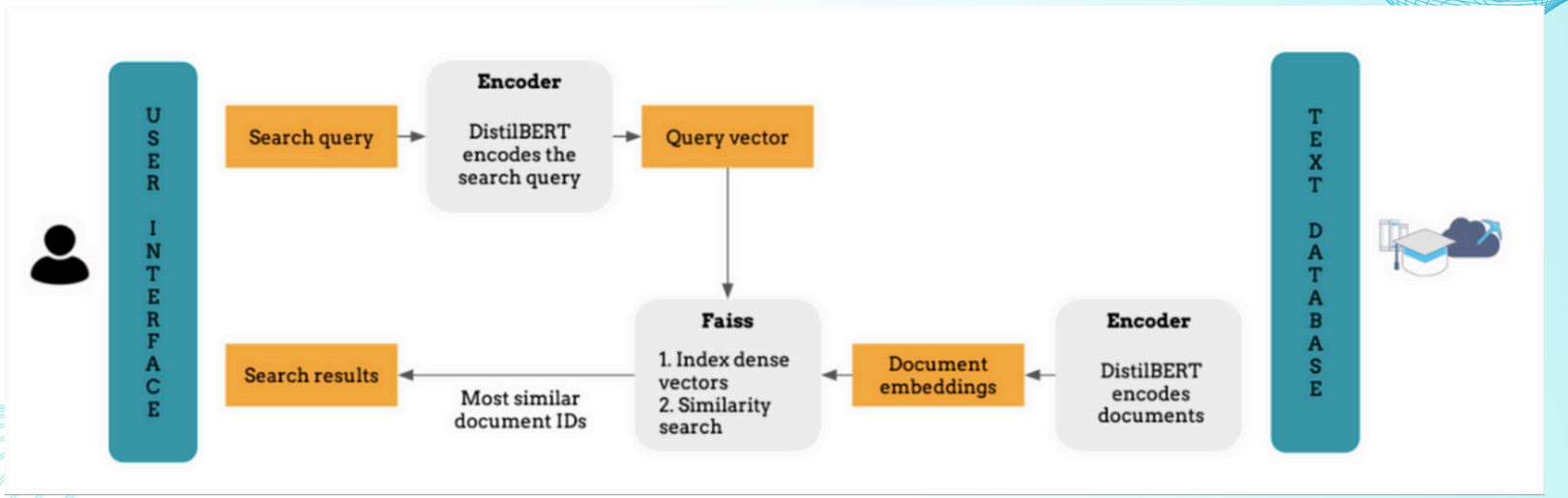
1.) Bi-encoders(Retrieval) + Cross encoders(Rerank)
2.) SBERT with faiss(HNSW) indexing(retrieve) + cross-encoders(rerank)
3.) SBERT with faiss indexing(retrieve) + cross-encoders(rerank)

SPARSE AND DENSE RETRIVAL

$$\sum_{t \in q} \log\left(\frac{N}{df_t}\right) \cdot \frac{(k_1+1)\cdot tf_{t,d}}{k_1 \cdot \left((1-b)+b\cdot\frac{L_d}{L_{ave}}\right)+tf_{t,d}}$$

(a) Rank score between answer and query via SPARTA





1. BM25 Lexical Retrieval :

- Implemented BM25 model for word-level similarity in document and query matching.
- Tuned parameters k1 and b (0.9 and 0.4) for optimal document ranking.

2. Hybrid Sparse Retrieval (SPARTA) :

- Integrated SPARTA for two-stage sparse retrieval with transformer-based refinement.
- Utilized DistilBERT embeddings for similarity scoring.

3. SBERT for Dense Retrieval :

- Utilized SBERT (msmarco-distilbert-base-v3) for dense retrieval.
- - Found superior performance in pre-trained models compared to in-batch negatives in SBERT (distilbert-base-uncased) fine-tuning.

1. SBERT(Trained with In-Batch Negatives) :

- SBERT fine-tunes its embeddings using contrastive learning, where a query is trained to match its positive document and distinguish it from negatives.

2. SBERT(Pre-trained) :

- Uses a pre-trained SBERT model directly (without additional fine-tuning). Since pre-trained embeddings are good at capturing semantic meanings leading to a better performance.

3. SBERT with FAISS Indexing(HNSW) :

- Embeddings stored in a FAISS HNSW index are optimized for advanced nearest neighbor search, utilizing a multi-layered graph structure for efficient query results.

4. SBERT with FAISS Indexing(FLAT) :

- Compared with the HNSW counterpart this more accurate version works more in a brute force method by searching the embeddings linearly for better matches in exchange for speed.

5. Bi-Encoders(Retrieval) & Cross-Encoders(Rerank) :

- Implemented a two-step approach, employing bi-encoders for initial retrieval and cross-encoders for reranking, ensuring superior performance by considering both query and passage.

6. SBERT with FAISS Indexing(Retrieve) & Cross-Encoders(Rerank) :

- Used FAISS indexing for efficient cosine similarity search, employing SBERT for vector generation and cross-encoder reranking.

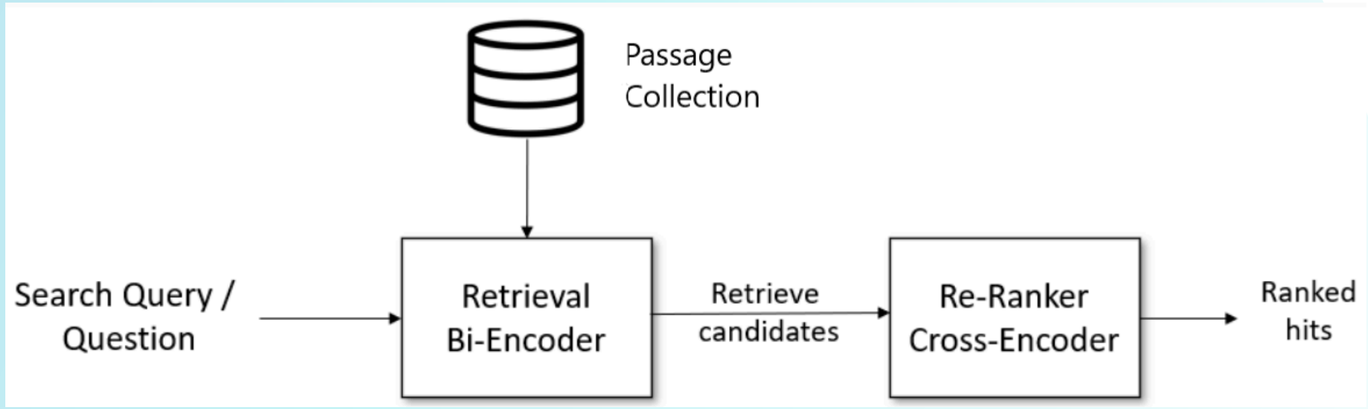
7. SBERT with FAISS(HNSW) Indexing(Retrieve) & Cross-Encoders(Rerank) :

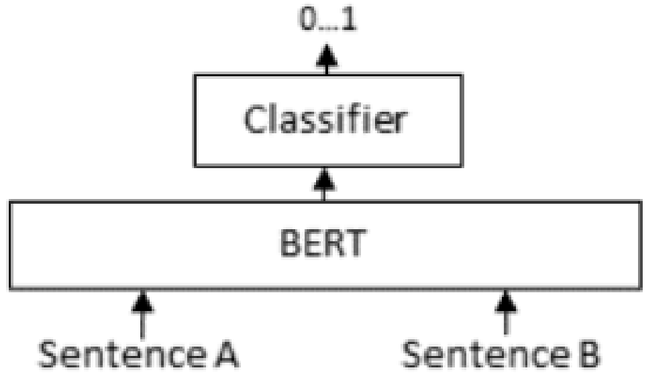
- Implemented HNSW indexing for advanced nearest neighbor search, utilizing a multi-layered graph structure for efficient query results.

8. Efficient Top-20 Reranking :

- Balanced efficiency and performance by bi-encoder retrieval of top 20 candidates, applying computationally intensive cross-encoder reranking solely on the top candidates.

BI-ENCODER(RETRIEVE) + CROSS ENCODER(RERANK)

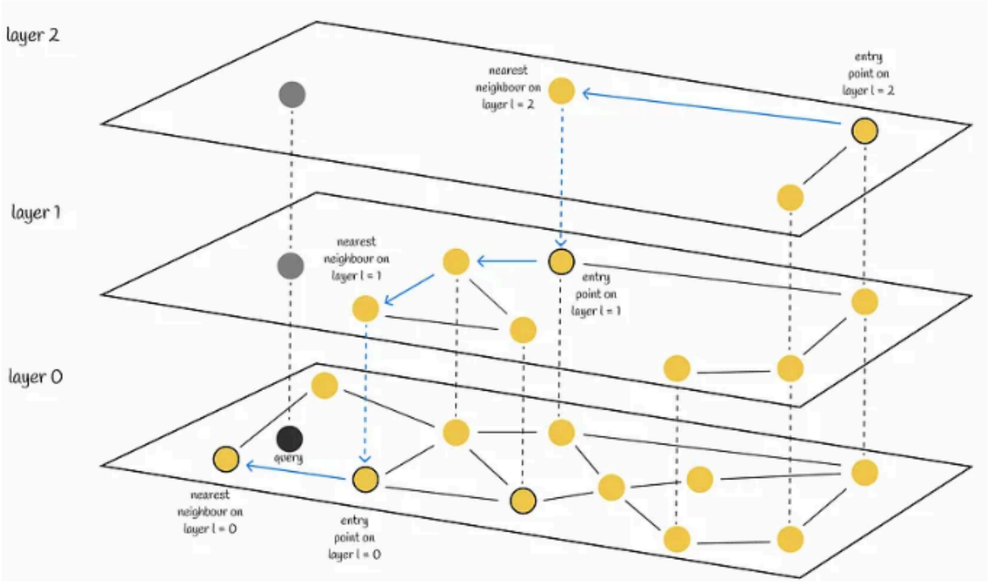


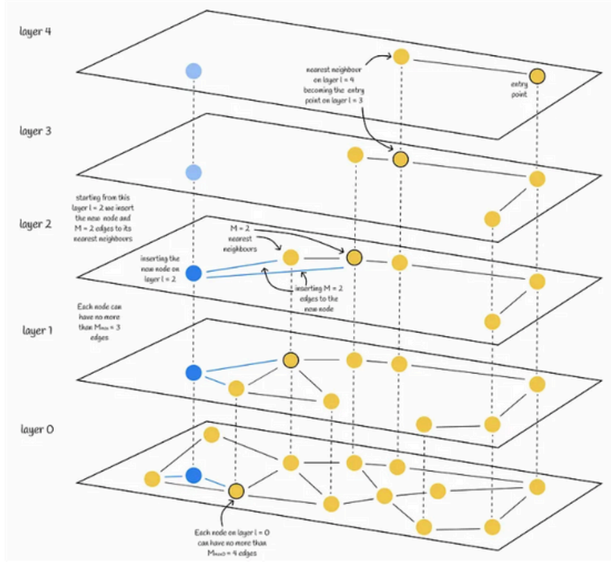


FAISS

Scalable Search With Facebook AI

SBERT WITH FAISS INDEXING(RETRIEVE) + CROSS-ENCODERS(RERANK)





Insertion of a node (in blue) in HNSW. The maximum layer for a new node was randomly chosen as l = 2. Therefore, the node will be inserted on layers 2, 1 and 0. On each of these layers, the node will be connected to its M = 2 nearest neighbours.

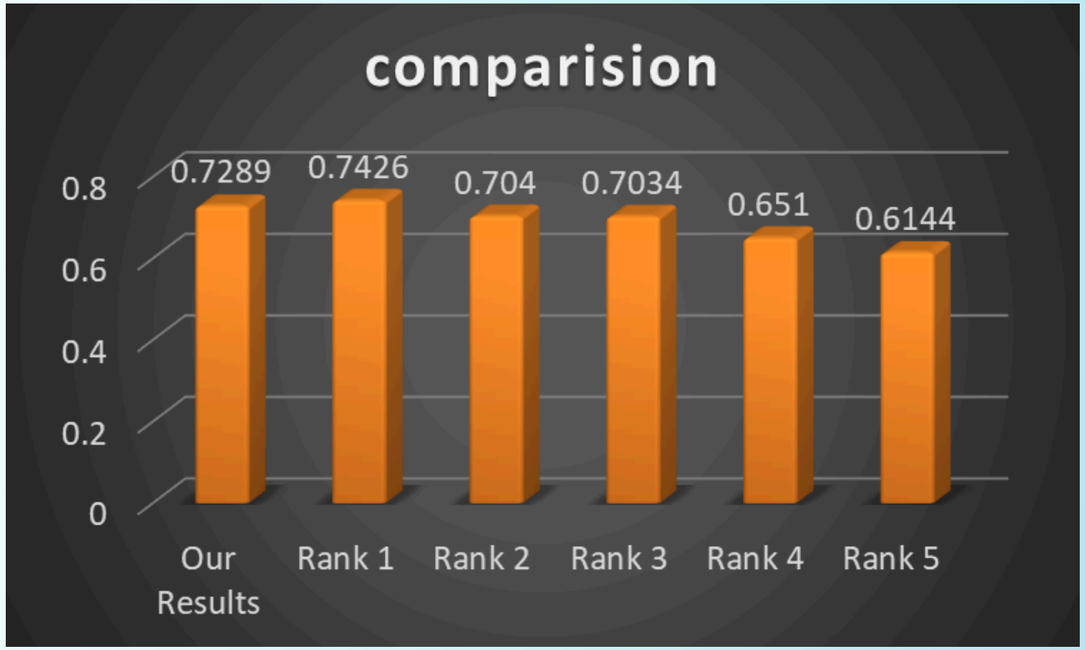
RESULTS

	Model	nDCG@10
0	BM25	0.3991
1	Elastic BM25	0.4768
2	SPARTA	0.5087
3	SBERT (trained with in-batch negatives)	0.6431
4	SBERT (pretrained)	0.693
5	SBERT with faiss indexing (HNSW)	0.703
6	SBERT with faiss (HNSW) indexing (retrieve) + cross-encoders (rerank)	0.7087
7	BiEncoder + Cross Encoder	0.7109
8	SBERT with faiss indexing	0.7209
9	SBERT with faiss indexing (IP)(retrieve) + cross-encoders (rerank)	0.7289

To exit

COMPARISION WITH TREC 2022 RESULTS

Model	nDCG@10
Our Results	**0.7289**
Rank 1	0.7426
Rank 2	0.704
Rank 3	0.7034
Rank 4	0.651
Rank 5	0.6144



OVERVIEW PAPER AND TOP PAPERS

- https://trec.nist.gov/pubs/trec31/papers/Overview_deep.pdf
- https://trec.nist.gov/pubs/trec31/papers/Ali.D.pdf
- https://arxiv.org/pdf/2208.07670.pdf
- https://trec.nist.gov/pubs/trec31/papers/CIP.D.pdf
- https://trec.nist.gov/pubs/trec31/papers/UoGTr.D.pdf
- https://trec.nist.gov/pubs/trec31/papers/yorku22.D.pdf