Better Retrieval for Generation

Ankit Saha - AI21BTECH11004
Pranav Balasubramanian - AI21BTECH11023

Problem Statement

What is Retrieval-Augmented Generation (RAG)?
To address the issue of Large Language Model (LLM) hallucinations and irrelevant outputs, we provide relevant information from a set of retrieved passages (related to the query) to provide the LLM with the necessary context to keep the answer grounded to the query being asked.

This project aims to develop better retrieval strategies that out-perform a set baseline in terms of retrieving.

MS MARCO Examples

query: what is rappelling

query_type: description

passages:
“Rappelling is the process of coming down from a mountain that is usually done with two pieces of rope. Use a natural anchor or a set of bolts to rappel from with help from an experienced rock climber in this free video on rappelling techniques. Part of the Video Series: Rappelling & Rock Climbing.",

……..

a descent of a vertical cliff or wall made by using a doubled rope that is fixed to a higher point and wrapped around the body. abseil. mountain climbing, mountaineering-the activity of climbing a mountain. descent-the act of changing your location in a downward direction."

answer: Rappelling is the process of coming down from a mountain that is usually done with two pieces of rope.

Retrieval Methodology

- The set of all passages is stored in a **FAISS vector database**
- An embedding model is used to first encode each passage
- The encodings are then indexed and stored
- We use the **IndexFlatIP** index, which indexes embeddings based on cosine similarity
- Given a query, encode it using the same embedding model
- Find its nearest neighbours efficiently in the embedding space using the index
- These correspond to the passages with the highest **cosine similarity** with the query, and thus form our retrieved context (most relevant passages to the query)

Embedding Models Tested

- For each query, there are relevant passages with either zero or one passage out of them marked, denoting the passage that was used to generate the answer
- We have tested the following embedding models and measured their top-1 and top-3 accuracies in terms of identifying the marked passage out of all the passages
- The marked passage can be thought of as the best passage among the relevant passages
- The following accuracies have been reported for a part of the training dataset because these marked labels are not present in the test dataset

Embedding Models Tested (contd.)

Model name	Top-1 accuracy	Top-3 accuracy
distilbert-base-uncased	12.8248	36.7980
bert-base-uncased	9.3880	28.1643
all-MiniLM-L6-v2	**40.4023**	**75.3562**
bge-large-en-v1.5	44.8449	78.2900
ms-marco-MiniLM-L-12-v2 *(cross-encoder)*	49.7904	83.7385

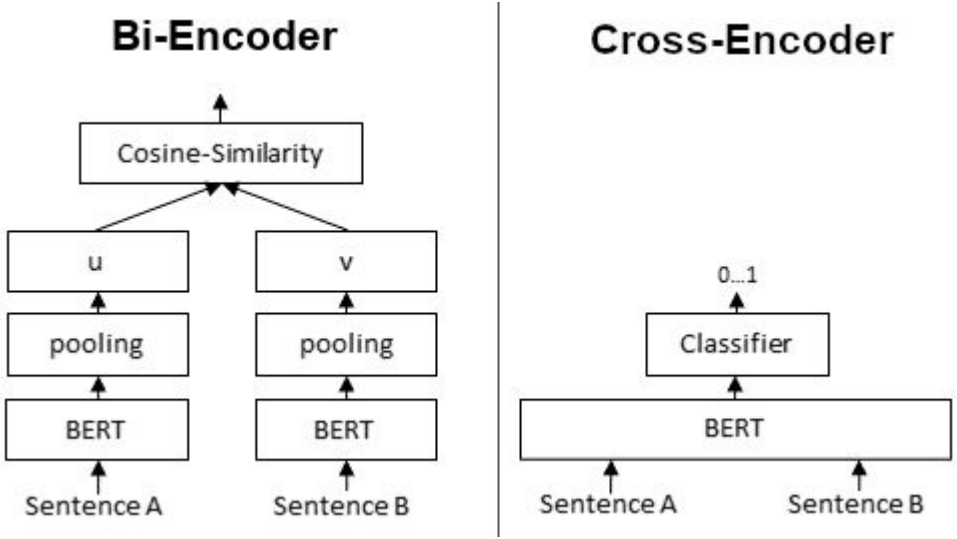
Baseline

- We have chosen the sentence transformer **all-MiniLM-L6-v2** as the baseline because
  - Its embeddings have been optimized for tasks like clustering and semantic similarity unlike BERT and DistilBERT
  - It is fairly lightweight (22.7M parameters) enabling fast fine-tuning and embedding of all passages in the vector database
- For evaluating the baseline, we have indexed all the 1,010,920 test dataset passages in the vector database and retrieved the top-K passages and computed the metrics mentioned in our first presentation for various values of K
- The metrics have been computed considering the relevant passages for each query as the true positives

Baseline Metrics

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.7077	0.1719	0.5706
5	0.6983	0.2546	0.5072
10	0.6591	0.3824	0.3812
100	0.5074	0.6832	0.0681

Improving the Baseline: Reranking



Cross-encoders do not give embeddings!

Baseline + Reranker

- We first use the encoder baseline to fetch the 100 closest passages (out of 1,010,920) from the FAISS index
- Then, we use **ms-marco-MiniLM-L-6-v2**, a cross-encoder model, which was trained on the MS-MARCO passage ranking dataset (different from our question-answering dataset)
- The cross-encoder is used to rerank the top 100 passages based on the similarity score with the query, which is then used for retrieving the best (top-K) set of passages
- There is a significant improvement observed in all the metrics

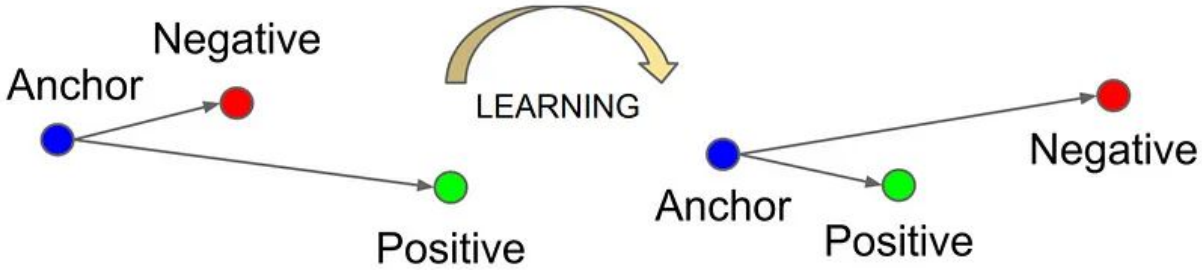
Baseline + Reranker Metrics

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.7564	0.1868	0.6202
5	0.7457	0.2805	0.5584
10	0.7045	0.4253	0.4237
100	0.5768	0.6832	0.0681

Improving the Baseline: Fine-Tuning (Initial Trials)

- We have then fine-tuned the baseline retriever **all-MiniLM-L6-v2** on our MS-MARCO dataset
- We have used the **TripletMarginWithDistanceLoss** as the loss function with the distance as the cosine distance
- Essentially, the loss takes a triplet as input: (anchor, positive, negative)
- The model will try to minimize the distance between the anchor (query) and the positive (relevant passage) and maximize the distance between the anchor (query) and the negative (irrelevant passage)

Triplet Margin With Distance Loss



$$\frac{1}{n}\sum_{i=1}^{n}\max\{d(a_i,p_i)-d(a_i,n_i)+\text{margin},0\}$$

Fine-tuning the Retriever (PyTorch)

- We have used all the 10 relevant passages for each query as the positives to form the triplets
- For negatives, we have randomly sampled passages corresponding to other queries
- The margin has been set as 1.0
- Adam optimizer with a learning rate of 1e-3 has been used
- 128,000 triplets were used for fine-tuning as of now

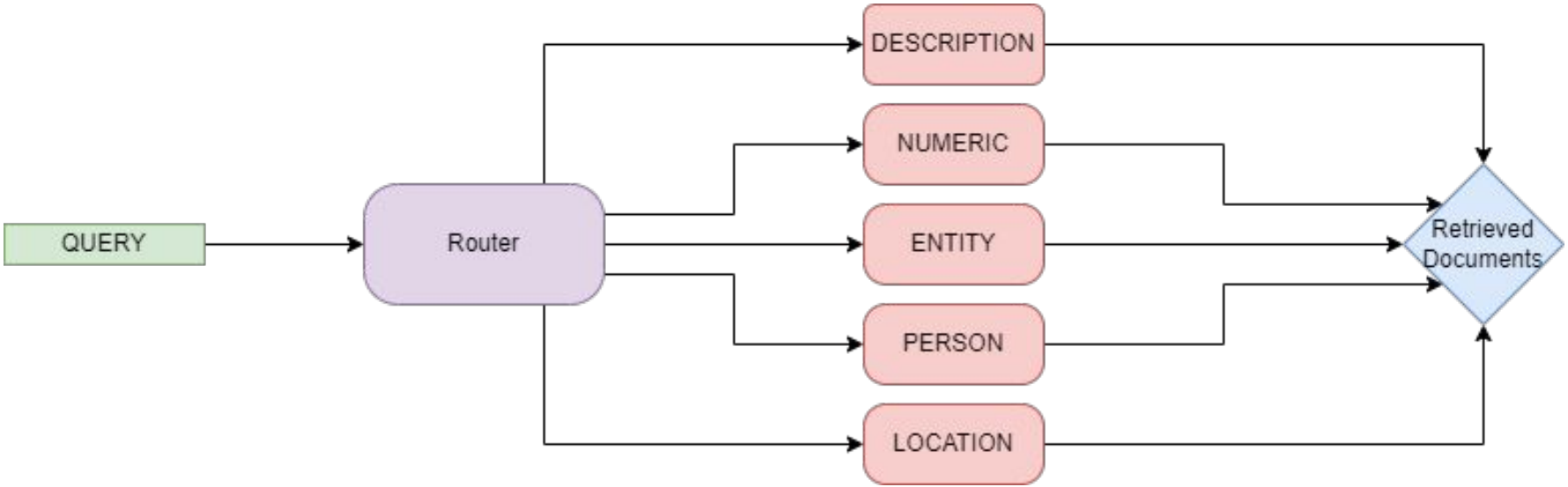
Fine-tuned + Reranker Metrics (PyTorch)

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.0006	0.00006	0.0002
5	0.0006	0.00006	0.0001
10	0.0006	0.00006	0.00006
100	0.0006	0.00008	0.000008

- TripletLoss is highly sensitive to the choice of positive and negative samples, could be the reason for the above

Mixture of Experts

Mixture of experts

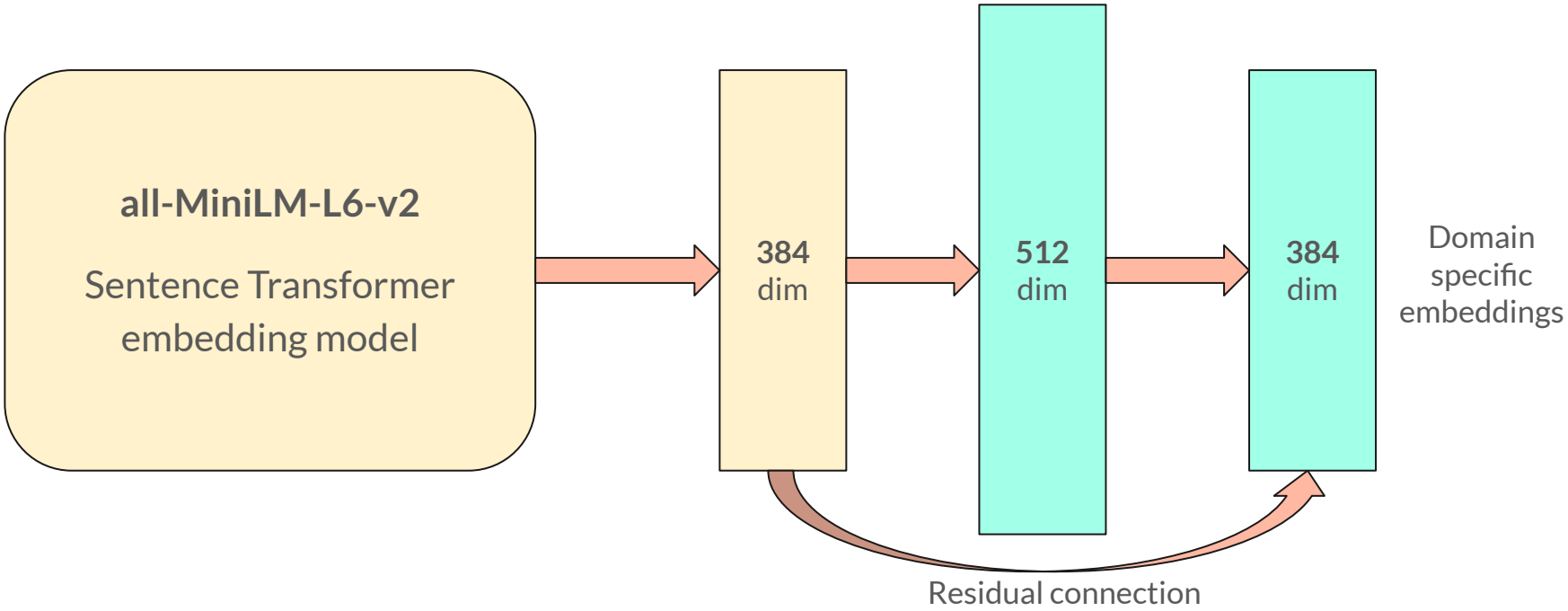


Adapter Fine-Tuning (LlamaIndex)

- The objective here is to freeze the embedding model, train an adapter (a smaller network) which transforms the current embedding space to another latent space (the domain specific space).
- The adapter chosen is a **TwoLayerNN**, from the **llama_index** library used with the following parameters :

```
{
        "in_features": 384,
        "hidden_features": 512,
        "out_features": 384,
        "bias": true,
        "activation_fn_str": "relu", "add_residual":
        true
}
```

Architecture of an Expert



Adapter Fine-Tuning (contd.)

- We then initialize the **EmbeddingAdapterFinetuneEngine**, an object to perform the fine tuning.
- The fine tuning engine uses **MyMultipleNegativesRankingLoss**, which is similar to the **MultipleNegativesRankingLoss** in the Sentence Transformers library, but optimized for llama_index embeddings
- Each query type's retriever was fine-tuned for 10 epochs
- The number of examples used for each **query_type** for fine tuning is as shown below:

```
NUM_EXAMPLES = {
    'DESCRIPTION': 100000,
    'NUMERIC': 100000,
    'ENTITY': 63243,
    'LOCATION': 46879,
    'PERSON': 42800
}
```

Multiple Negative Rankings Loss

- Very similar to the Triplet Loss mentioned earlier
- The difference is that this loss only requires positive passages for each query
- Minimizing the loss is equivalent to bringing the positive passage embeddings closer to the query embeddings while taking the embeddings for passages corresponding to other queries farther away

The Router

- The router is a classification network which forwards the incoming query to the best retriever.
- Each query is classified into one of the five possible query types.
- Was trained on ‘train-00005-of-00007.parquet’ , this was chosen since it had an almost equal portion of all query types which would help the router to distinguish between the query type.
- The input to the network is the baseline embedding of the query by the sentence transformer model.
- Loss function used: torch.nn.CrossEntropyLoss()

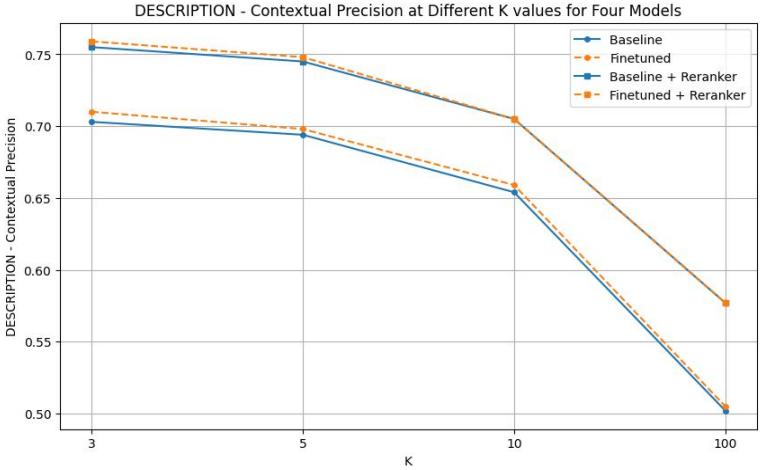
Router performance

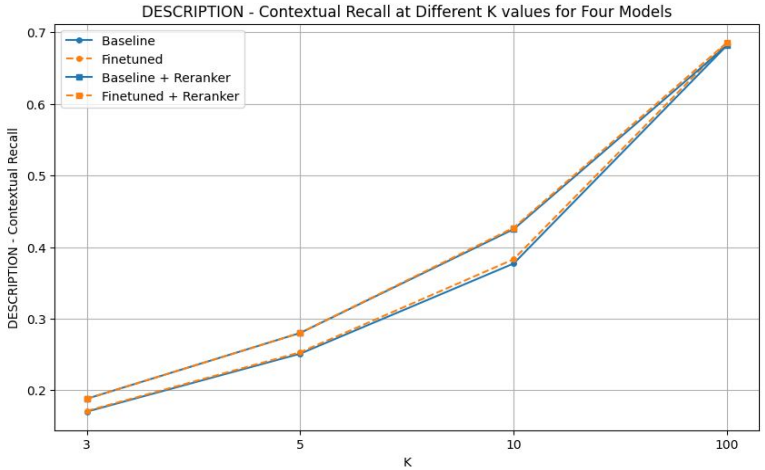
- input_dim = 384 (dimensionality of embedding)
- output_dim = 5 (number of query types)

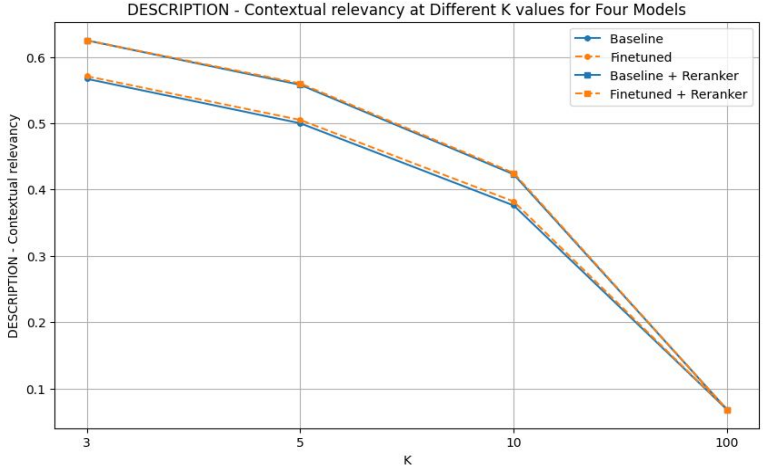
- With 128 hidden dimensions, trained over 70 epochs:
  Accuracy of router: **0.8837**
- With 256 hidden dimensions, trained over 70 epochs:
  Accuracy of router: **0.8882**
- With 256 hidden dimensions, trained over 120 epochs:
  Accuracy of router: **0.8928**

Comparison of models

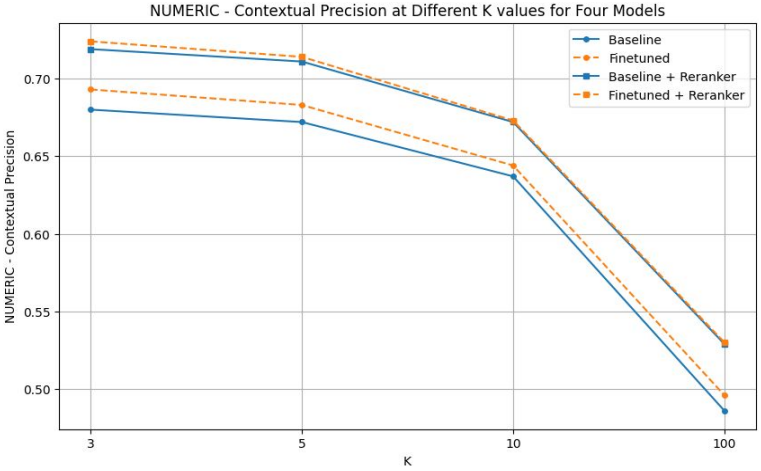
DESCRIPTION Queries

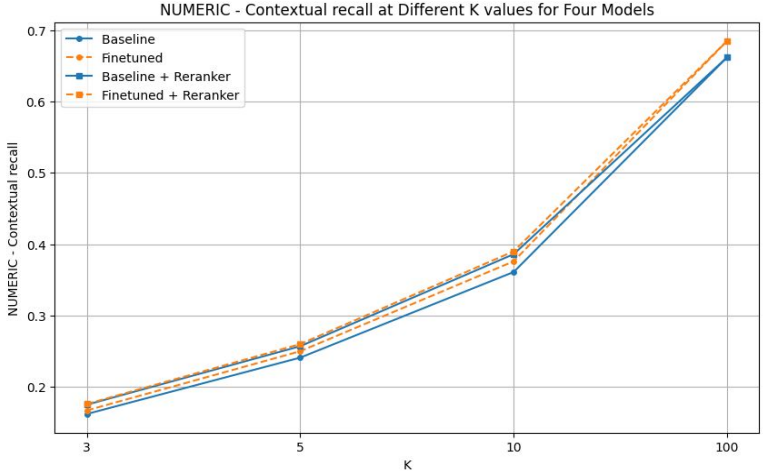


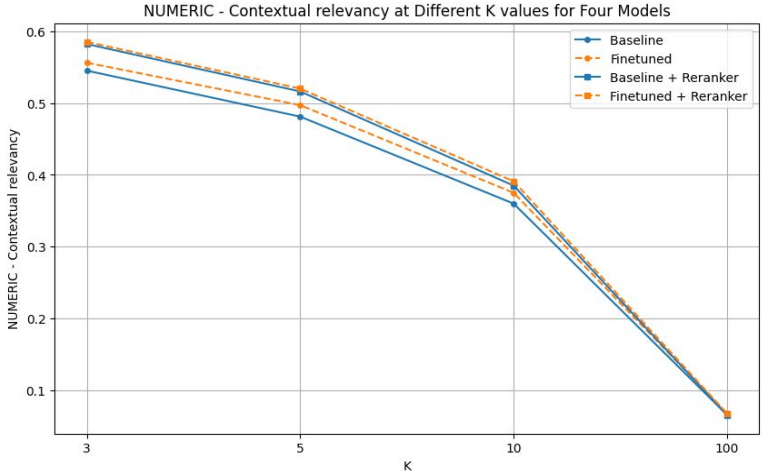




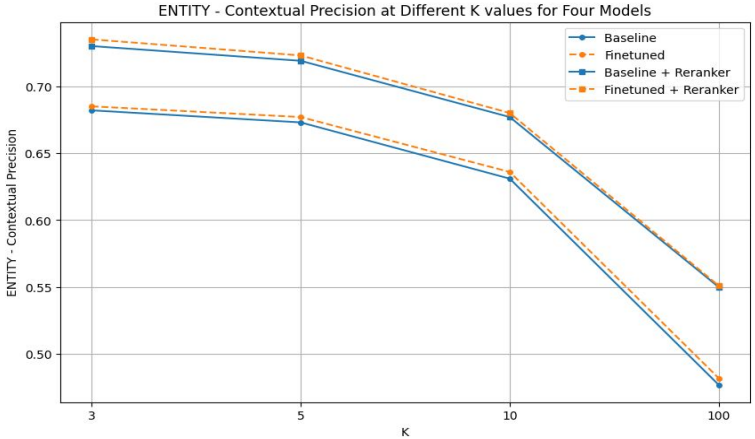
NUMERIC Queries

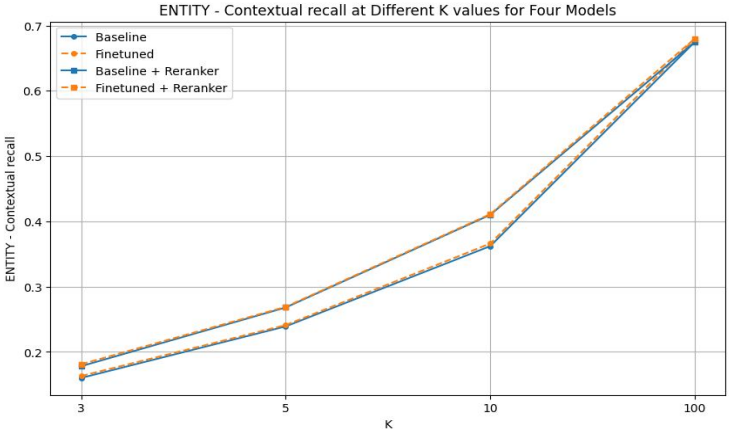


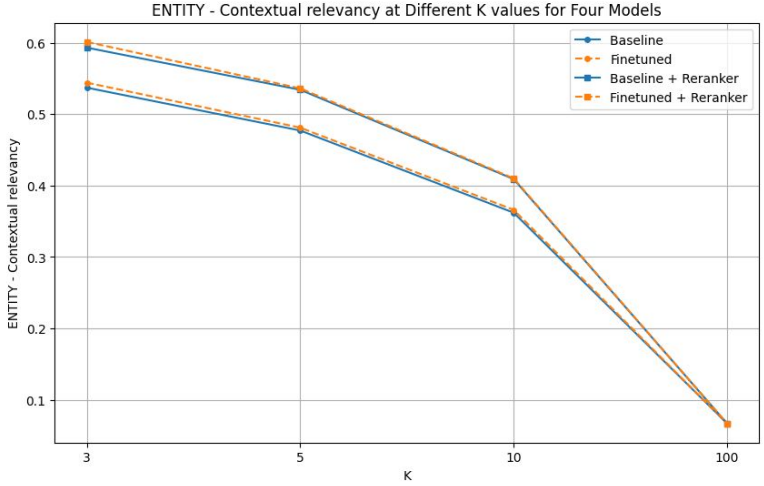




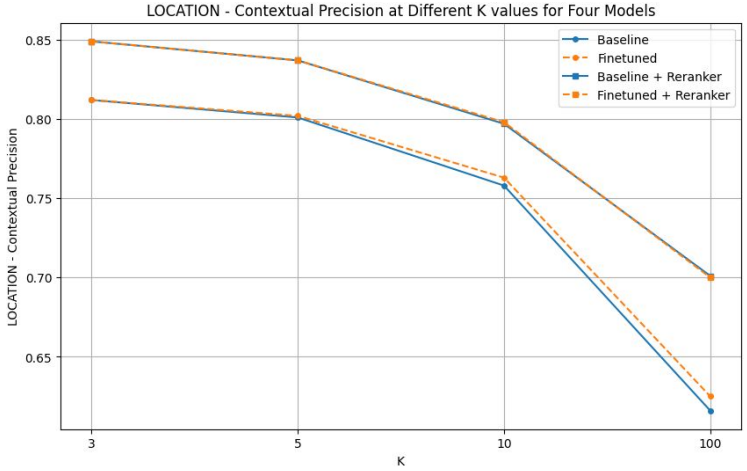
ENTITY Queries

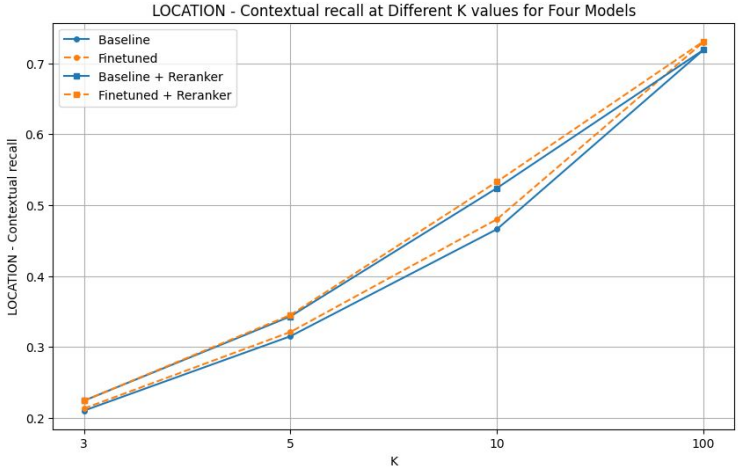


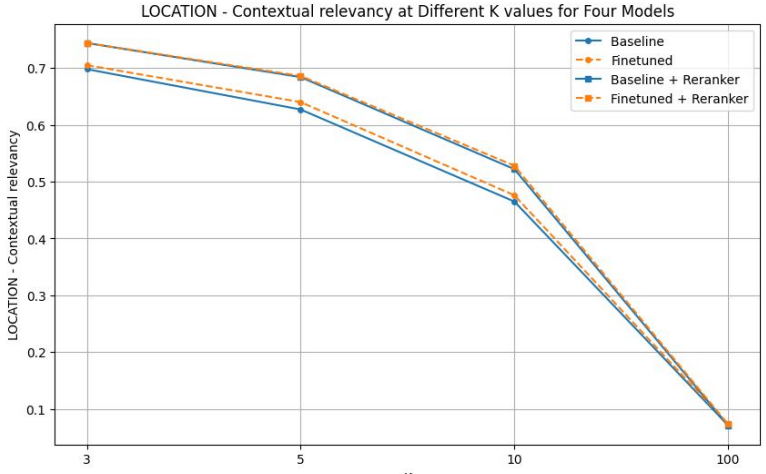




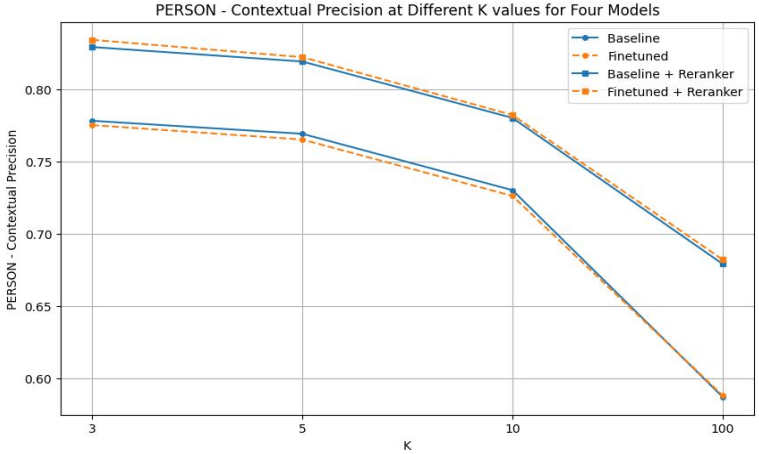
LOCATION Queries

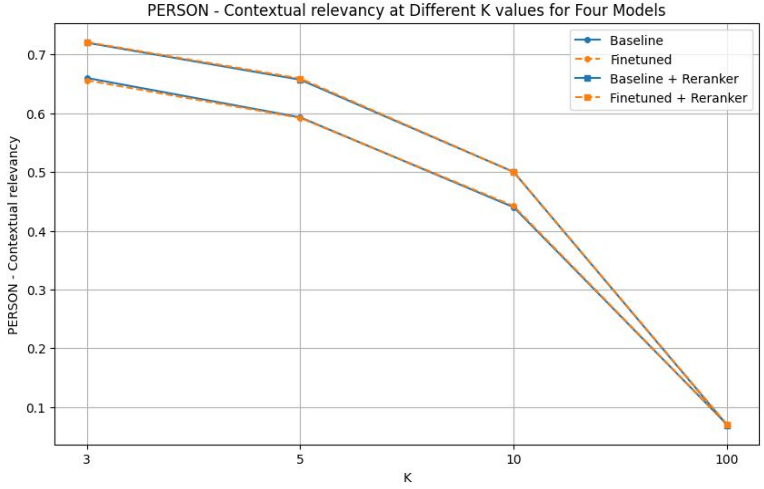




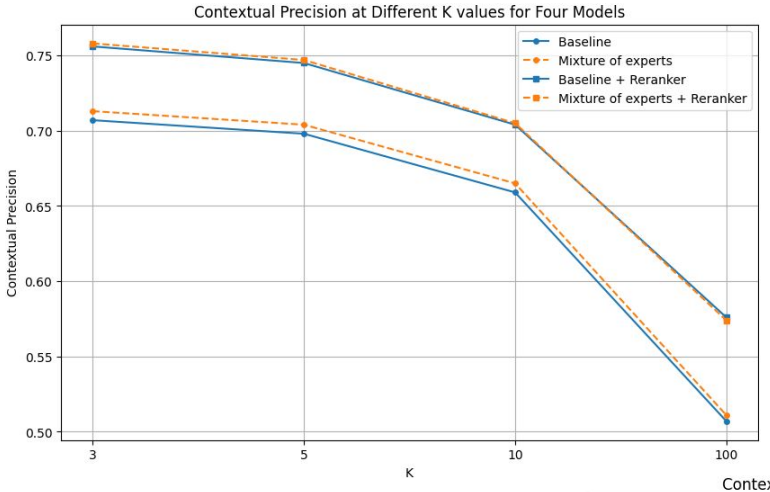


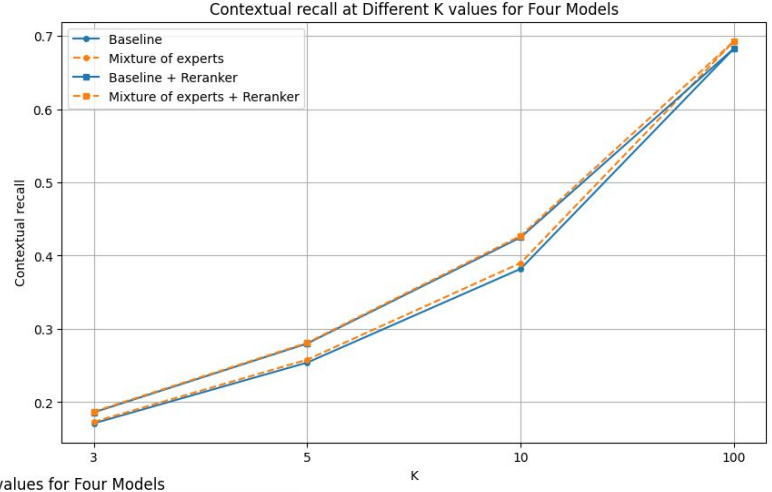
PERSON Queries

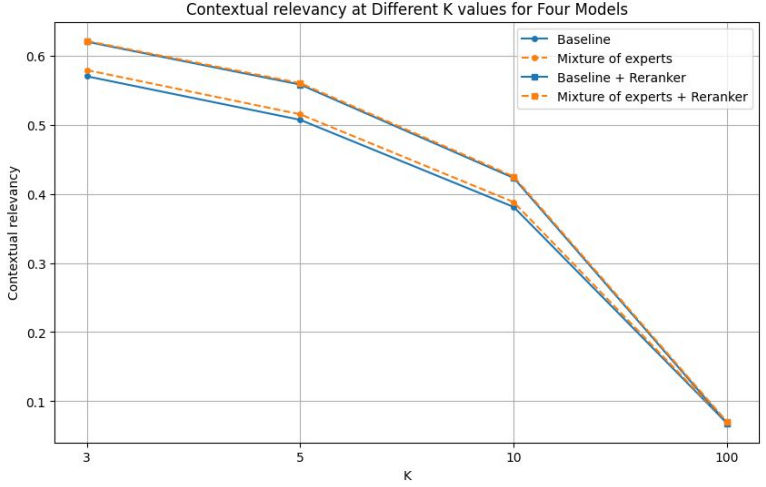




Mixture of Experts' Performance





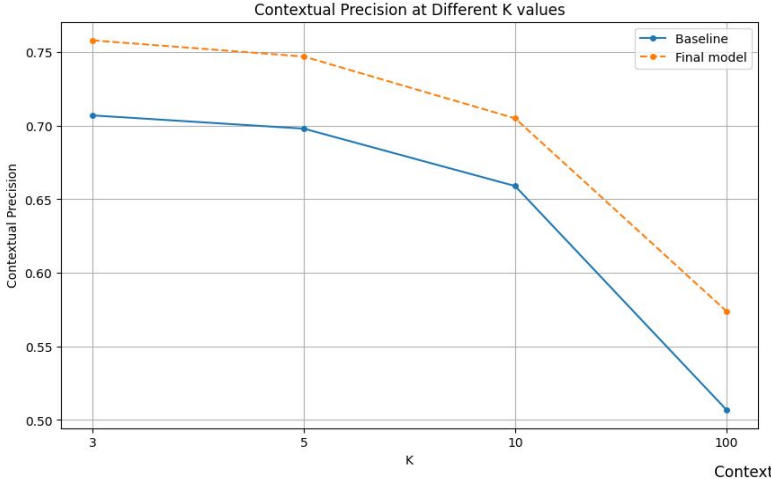


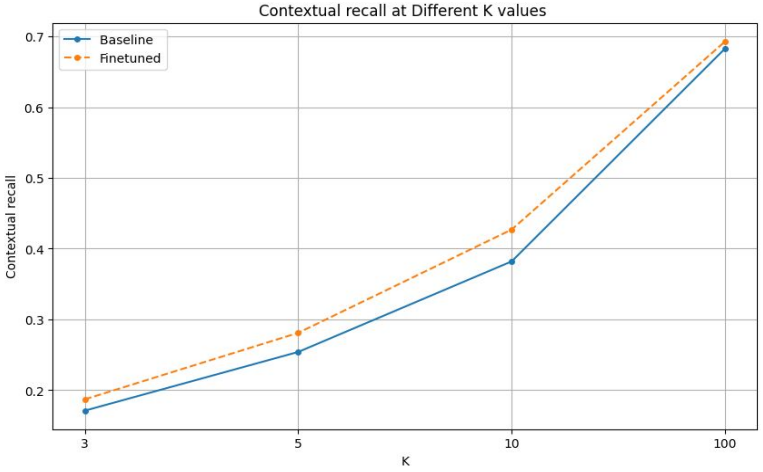
Summary and Conclusions

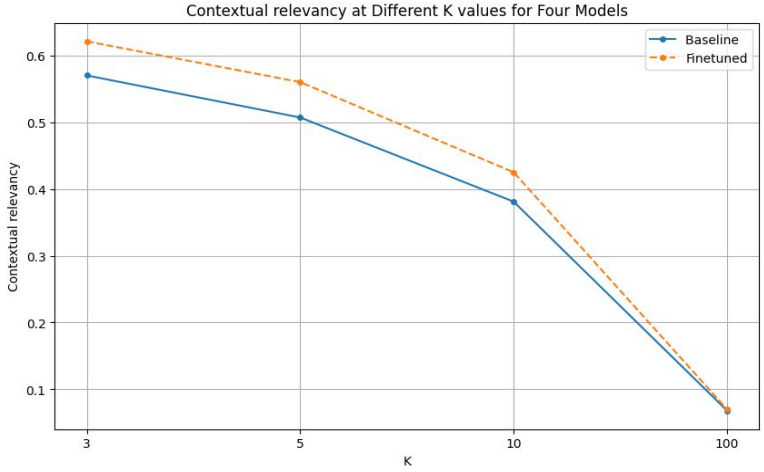
Summary of our Approach

- We have used the **all-MiniLM-L6-v2** sentence transformer as the baseline embedding model
- For each of the five query types, we fine-tuned separate **expert retrievers** that specialize in that particular domain
- A **vector database** is constructed for each expert
- A **router** is trained which given a query, outputs the query type
- The query is then forwarded to the appropriate expert, which then retrieves a set of K most relevant passages
- Finally, the set of passages is reranked using the **ms-marco-MiniLM-L-6-v2** cross-encoder

Final model vs Baseline







References

- Sentence Transformers
- Cross-Encoders
- Intution of Triplet Loss
- TripletMarginWithDistanceLoss
- Loss Functions
- LlamaIndex Fine-Tuning
- Mixtral of Experts

Appendix

DESCRIPTION

Fine-Tuned

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.710	0.171	0.571
5	0.698	0.253	0.505
10	0.659	0.383	0.382
100	0.505	0.686	0.068

Baseline

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.703	0.170	0.567
5	0.694	0.251	0.500
10	0.654	0.377	0.376
100	0.502	0.682	0.068

DESCRIPTION

Fine-Tuned with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.759	0.188	0.625
5	0.748	0.280	0.560
10	0.705	0.427	0.425
100	0.577	0.686	0.068

Baseline with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.755	0.188	0.625
5	0.745	0.280	0.558
10	0.705	0.425	0.423
100	0.577	0.682	0.068

NUMERIC

Fine-Tuned

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.693	0.167	0.556
5	0.683	0.250	0.497
10	0.644	0.376	0.375
100	0.496	0.685	0.068

Baseline

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.680	0.162	0.545
5	0.672	0.241	0.481
10	0.637	0.361	0.360
100	0.486	0.662	0.066

NUMERIC

Fine-Tuned with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.724	0.176	0.585
5	0.714	0.260	0.520
10	0.673	0.390	0.391
100	0.530	0.685	0.068

Baseline with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.719	0.175	0.582
5	0.711	0.257	0.516
10	0.672	0.386	0.385
100	0.529	0.662	0.066

ENTITY

Fine-Tuned

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.685	0.163	0.544
5	0.677	0.241	0.481
10	0.636	0.366	0.366
100	0.482	0.680	0.067

Baseline

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.682	0.160	0.537
5	0.673	0.239	0.477
10	0.631	0.362	0.362
100	0.477	0.675	0.067

ENTITY

Fine-Tuned with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.735	0.181	0.601
5	0.723	0.269	0.536
10	0.680	0.411	0.410
100	0.551	0.680	0.067

Baseline with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.730	0.178	0.593
5	0.719	0.268	0.534
10	0.677	0.410	0.409
100	0.550	0.675	0.067

LOCATION

Fine-Tuned

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.812	0.213	0.705
5	0.802	0.321	0.640
10	0.763	0.480	0.476
100	0.625	0.730	0.073

Baseline

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.812	0.210	0.698
5	0.801	0.315	0.627
10	0.758	0.466	0.465
100	0.616	0.719	0.071

LOCATION

Fine-Tuned with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.849	0.224	0.744
5	0.837	0.345	0.686
10	0.798	0.533	0.528
100	0.700	0.731	0.073

Baseline with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.849	0.224	0.744
5	0.837	0.343	0.684
10	0.797	0.524	0.522
100	0.701	0.719	0.071

PERSON

Fine-Tuned

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.775	0.198	0.656
5	0.765	0.297	0.592
10	0.726	0.443	0.442
100	0.588	0.703	0.070

Baseline

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.778	0.199	0.660
5	0.769	0.298	0.593
10	0.730	0.442	0.440
100	0.587	0.701	0.070

PERSON

Fine-Tuned with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.834	0.217	0.721
5	0.822	0.330	0.659
10	0.782	0.503	0.500
100	0.682	0.703	0.070

Baseline with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.829	0.217	0.720
5	0.819	0.330	0.657
10	0.780	0.502	0.500
100	0.679	0.701	0.069

Mixture of experts vs Baseline

Mixture of experts

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.713	0.173	0.579
5	0.704	0.258	0.515
10	0.665	0.390	0.388
100	0.511	0.693	0.069

Baseline

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.707	0.171	0.570
5	0.698	0.254	0.507
10	0.659	0.382	0.381
100	0.507	0.683	0.068

Mixture of experts vs Baseline

Mixture of experts with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.758	0.187	0.621
5	0.747	0.281	0.560
10	0.705	0.427	0.425
100	0.574	0.693	0.070

Baseline with Reranker

K	Contextual Precision	Contextual Recall	Contextual Relevancy
3	0.756	0.186	0.620
5	0.745	0.280	0.558
10	0.704	0.425	0.423
100	0.576	0.683	0.068