Problem Statement

In the highly specialized medical equipment industry, delivering exceptional customer service, particularly in sales and technical support, is crucial for securing sales and fostering long-term customer loyalty. However, the complexity of medical equipment can make it challenging for customers to obtain detailed product information, compatibility guidance, and technical troubleshooting assistance. This project aimed to develop a Question Answering (QA) Information Retrieval System capable of retrieving relevant documents to address customer inquiries about a medical technology company's product catalog, thereby enhancing customer support and streamlining the sales process.

Data Wrangling

Dataset Description

To prepare the data for the QA Information Retrieval System, two CSV files were created from the company's product offering catalog: `probes.csv` and `systems.csv`.

The `probes.csv` file contained the following information about each ultrasound probe:

- Manufacturer
- Probe model
- Cartridge connection availability
- Compatible ultrasound systems
- Array type (e.g., convex, linear)
- Applications (e.g., abdominal, obstetric)
- Operational frequency range (in MHz)
- Current stock level

The `systems.csv` file included the following information about each ultrasound system:

- Ultrasound system model
- Manufacturer
- List of compatible probes

Converting Tabular Data into Text Documents

To convert the tabular data into a format suitable for the QA system, each row, column, and cell was transcribed into human-readable sentences containing the key information. Custom Python functions were applied to each row of the probes and systems DataFrames, resulting in multiple documents for each probe/system. Each document provided a sentence of information that

could be extrapolated from the original .csv file, enhancing compatibility with natural language processing techniques and improving readability for the QA system.

Document examples:

```
"The manufacturer of the C5-2 probe is ATL."
"The ATL C5-2 probe is compatible with the following systems: HDI 1500, HDI 3000, HDI 3500, HDI 5000."
"The ATL C5-2 is a convex array type probe."
"The ATL C5-2 probe is suitable for the following applications: abdominal, general."
"The ATL C5-2 probe is currently in stock and available for sale.
The frequency range of the ATL C5-2 probe is 2.0 - 5.0 MHz."
```

Synthetic Question Generation Using a Large Language Model

To generate a diverse set of questions for training and evaluating the QA Information Retrieval System, a large language model (LLM), llama3, was utilized. The goal was to create synthetic questions that could be answered based on the information provided in each document, improving the system's robustness and ability to handle a wide range of user queries.

To guide the model, a prompt template was crafted that provided examples of generating concise, answerable questions based on the information within each document. The model was instructed to generate three questions per document, following the format of the provided examples.

To streamline the process of extracting the questions from the model's output, a template was incorporated into the prompt, instructing the model to respond solely with the questions structured as a numbered list. This approach accounted for the variability in the language model's responses, as it was observed that llama3 doesn't always perfectly follow directions and is still susceptible to hallucinations for this task. However, as long as it remembered to number each generated question, regular expressions in Python could be utilized to extract only the questions, filtering out any unnecessary text.

Prompt snippet used for question generation:

```
Generate three concise questions that can be answered using the following information, similar to the
example provided. Provide only the questions, numbered as follows:

1. [insert question here]
2. [insert question here]
3. [insert question here]

Example 1:
Information: The manufacturer of the C3 probe is ATL.
Questions:
1. Who is the manufacturer of the C3 probe?
2. Who makes the C3 transducer?
3. Is the C3 probe made by ATL?
```

A regular expression function was created to look for a sequence of a number followed by a period, whitespace, and then capture the text until the first question mark. This allowed us to parse the outputs, map each question onto its corresponding document and tags, and create a training dataset containing question-document pairs.

Filtering Generated Questions for Relevance

To ensure the quality of the generated questions, another LLM prompt was employed to score each question-document pair based on its relevance. The prompt assessed whether a question could be answered using the information provided in the associated document. Any question-document pairs deemed irrelevant were manually reviewed, and the scoring was re-evaluated if necessary. These pairs were then merged back into the final dataset.

The final fine-tuning dataset comprised **963 queries** and **324 documents** in the corpus, serving as the foundation for training and evaluating the QA Information Retrieval System.

Exploratory Data Analysis

Evaluation of Pre-trained Embedding Models

In the next step of building the question-answering system, the focus was on selecting an appropriate embedding model. Embedding models convert text data into numerical representations, enabling the system to understand and compare the semantic meaning of words and phrases. The goal was to identify the best-performing out-of-the-box model, which would then be fine-tuned with the query-document dataset.

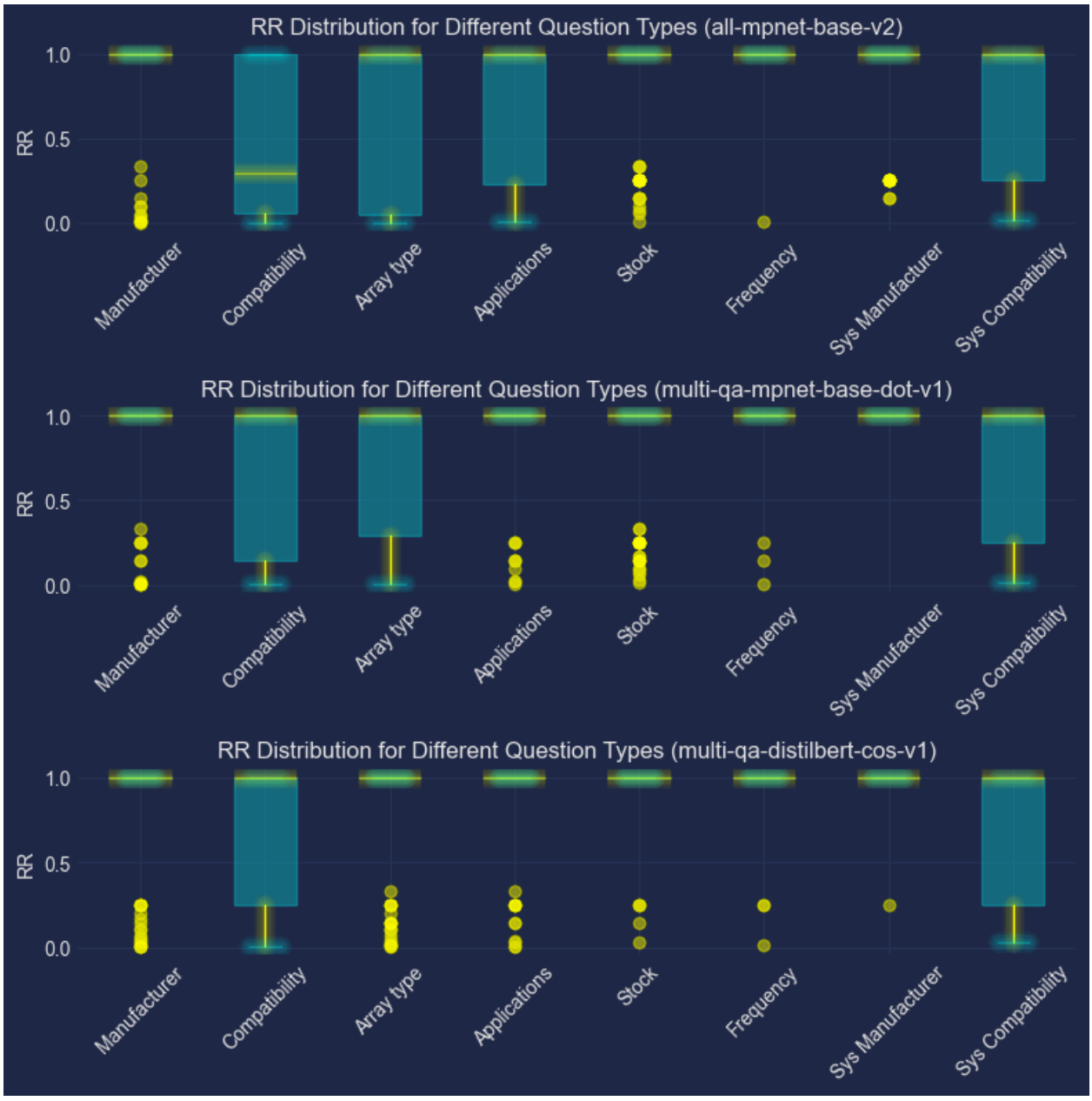
Three pre-trained models from Hugging Face were evaluated: *all-mpnet-base-v2*, *multi-qa-mpnet-base-dot-v1*, and *multi-qa-distilbert-cos-v1*. *all-mpnet-base-v2* is a general-purpose model, while the other two were specifically trained on a question-answer dataset of 215 million pairs. *multi-qa-distilbert-cos-v1* is based on the DistilBERT architecture, which uses knowledge distillation to create a smaller, faster model. *multi-qa-mpnet-base-dot-v1* utilizes the MPNet architecture, which employs masked and permuted language modeling.

To evaluate the models' performance, the **Mean Reciprocal Rank (MRR)** metric was used. Documents are ranked based on their cosine similarity scores with respect to the query embeddings, with more semantically similar documents receiving higher scores and ranks. MRR calculates the average reciprocal rank of the first relevant document retrieved for a set of queries, making it particularly useful when there is only one relevant document per query, as it focuses on the position of the first correct result.

Evaluation Results

The reciprocal rank distributions for each embedding model across eight question types (manufacturer, compatibility, array type, applications, stock, frequency range, system manufacturer, system compatibility) were visualized using box and whisker plots.

Reciprocal Rank Distribution by Question Type and Embedding Model

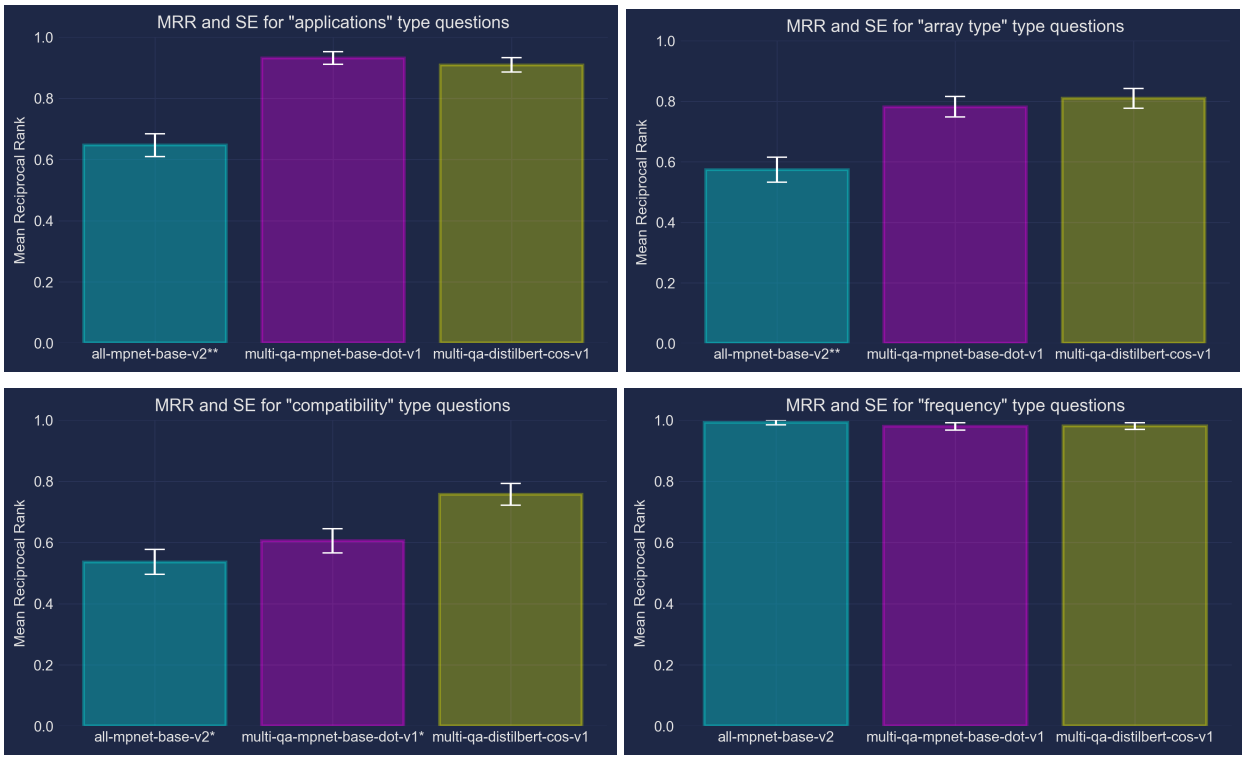


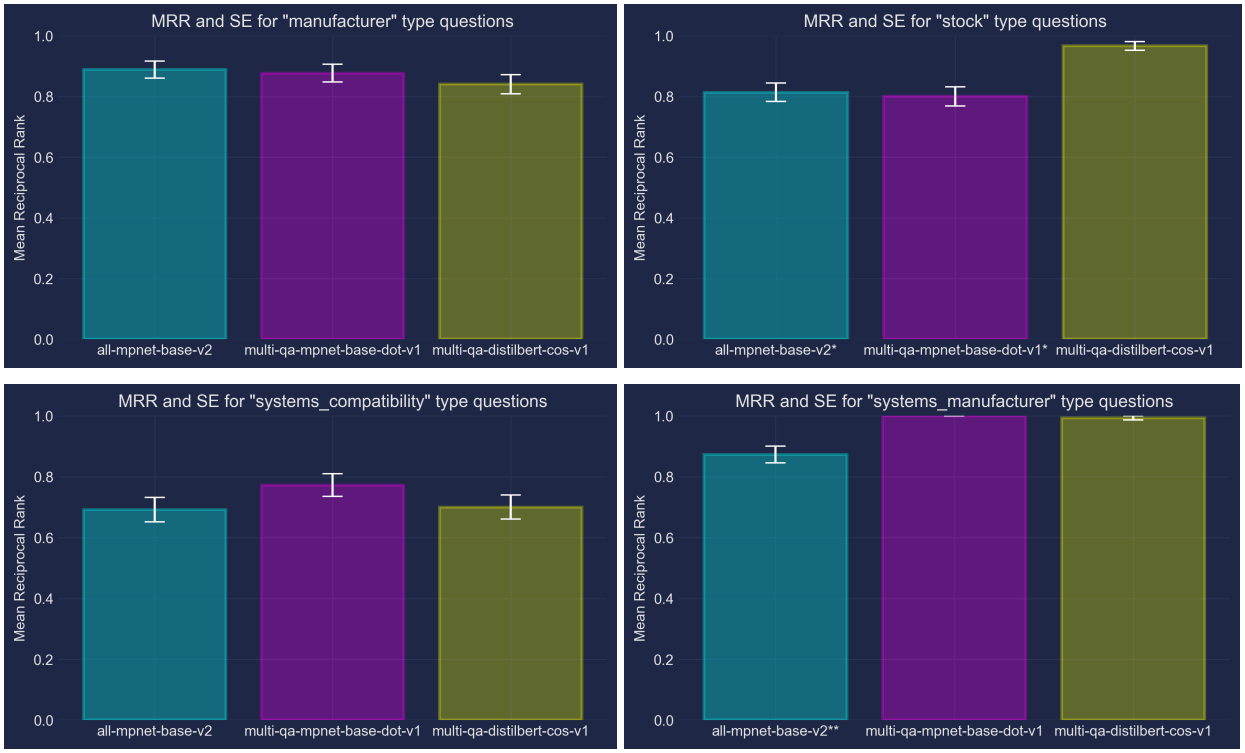
Upon examining the distribution of reciprocal rank scores, it was observed that the generalized model, *all-mpnet-base*, exhibited a higher spread in reciprocal rank scores across more question

type categories compared to the other two models. In contrast, both question-answering models, *multi-qa-distilbert-cos-v1* and *multi-qa-mpnet-base-dot-v1*, performed consistently well across all categories of questions. This is evident from their median reciprocal rank scores being 1 for all categories, indicating that the first relevant document was often ranked at the top position.

As part of the analysis, Tukey's Honest Significant Difference test was employed to assess the significance of performance differences between the embedding models. The test revealed that in 5 out of 8 question categories, the comparisons were statistically significant.

Mean Reciprocal Rank and SE across Various Question Categories





The results consistently highlighted a notable trend: the generalized model yielded statistically lower scores across multiple categories compared to the two QA models. Furthermore, the *multi-qa-distilbert-cos-v1* model demonstrated impressive performance, consistently achieving equally high, if not the highest, mean reciprocal rank scores across all question types.

These findings guided the decision-making process for selecting the most suitable embedding model. The *multi-qa-distilbert-cos-v1* model was chosen for the question-answering system, and efforts were focused on fine-tuning and optimizing its performance.

Fine-Tuning Process

Creating Datasets

Before splitting the data, a custom QADataset class was defined, which inherits from the PyTorch Dataset class. This class takes a list of InputExample objects, where each example consists of the query text and the content of the relevant document.

Three datasets were created for training, validation, and testing (70/15/15) by iterating over the respective query IDs and constructing InputExample objects with the query text and relevant document content.

Bayesian Optimization for Hyperparameter Tuning

Bayesian optimization was employed to find the optimal hyperparameters for fine-tuning the model. This approach aims to maximize the Mean Reciprocal Rank (MRR) metric on the validation set by intelligently searching the hyperparameter space.

The search space for the Bayesian optimization was defined as follows:

- per_gpu_batch_size: (16, 64)
- weight_decay: (0, 0.3)
- learning_rate: (1e-5, 5e-5)
- warmup_steps: (0, 500)
- num_epochs: (2, 5)

The BayesianOptimization library was used to perform the optimization, running for 15 iterations with 5 initial points. The best hyperparameters found were:

- per_gpu_batch_size: 56
- weight_decay: 0.21
- learning_rate: 3.6e-5
- warmup_steps: 106
- num_epochs: 4

Training the Final Model

With the best hyperparameters identified through Bayesian optimization, the next step was to train the final model using these optimal settings. DataLoader objects were created for the training, validation, and test datasets with the best batch size hyperparameter of 56. Several performance metrics were monitored throughout the 4 training epochs.

Loss Function

We utilized the `MultipleNegativesRankingLoss` function, which is well-suited for scenarios where we have only positive pairs of data, such as (query, relevant document) pairs. The loss function operates as follows:

1. Compute the cosine similarity scores between the query and document embeddings for each (query, document) pair in the batch.
2. Treat the actual (query, relevant document) pair as positive, and all other pairs in the batch as negatives.
3. Normalize the similarity scores using the softmax function, transforming them into a probability distribution.
4. Minimize the negative log-likelihood of these softmax-normalized scores, encouraging the model to assign higher probabilities to the relevant documents and lower probabilities to the irrelevant ones.

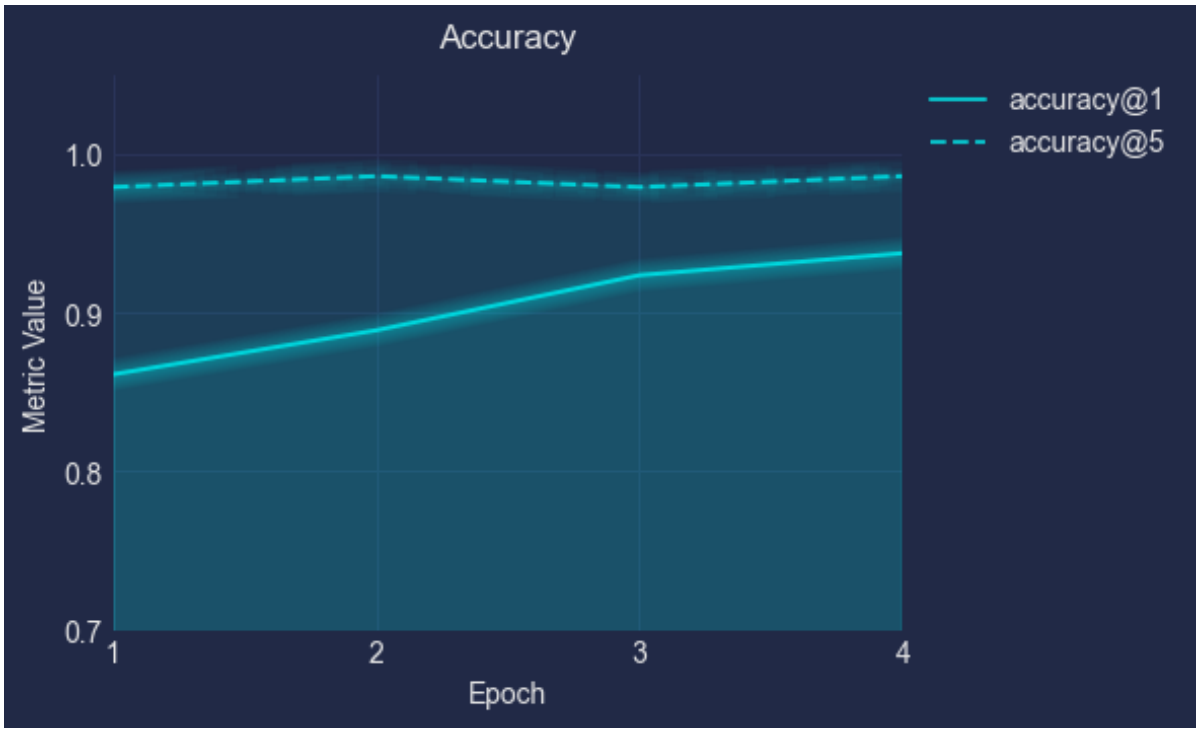
By minimizing this loss function during training, the model learns to rank the relevant documents higher than the irrelevant ones for a given query.

Evaluation Metrics

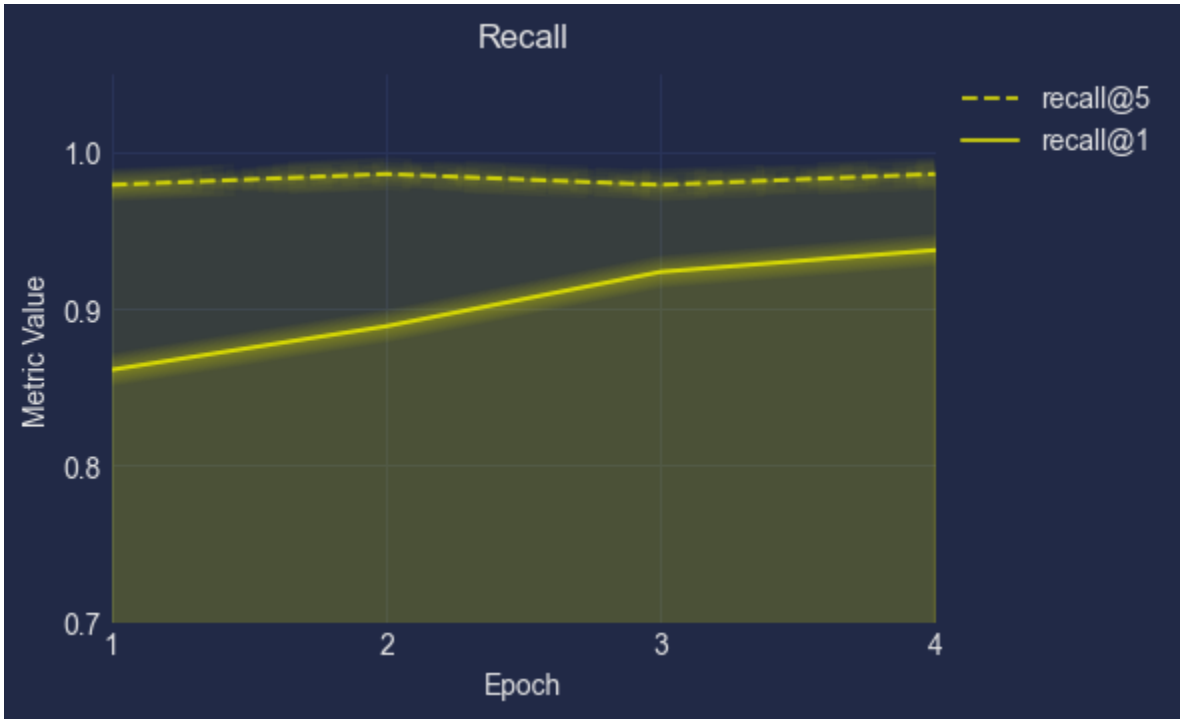
During the training process, we monitored several performance metrics to evaluate the effectiveness and progress of our model. These metrics were extracted from the training logs and visualized over the epochs. We tracked these metrics at different top-k values (e.g., @1, @5, @10) to understand how well the model performs at various levels of retrieval depth:

- **Accuracy@k:** The percentage of queries for which the relevant document is among the top-k retrieved documents.
- **Recall@k:** The percentage of relevant documents that were successfully retrieved among the top-k results.
- **Mean Reciprocal Rank (MRR)@k:** Average of the reciprocal ranks of the first relevant document for each query, considering only the top-k results. It provides a measure of how highly ranked the relevant documents are.
- **Normalized Discounted Cumulative Gain (NDCG)@k:** Measure of ranking quality that considers not only the relevance of the retrieved documents but also their positions in the ranked list, with more weight given to higher positions.

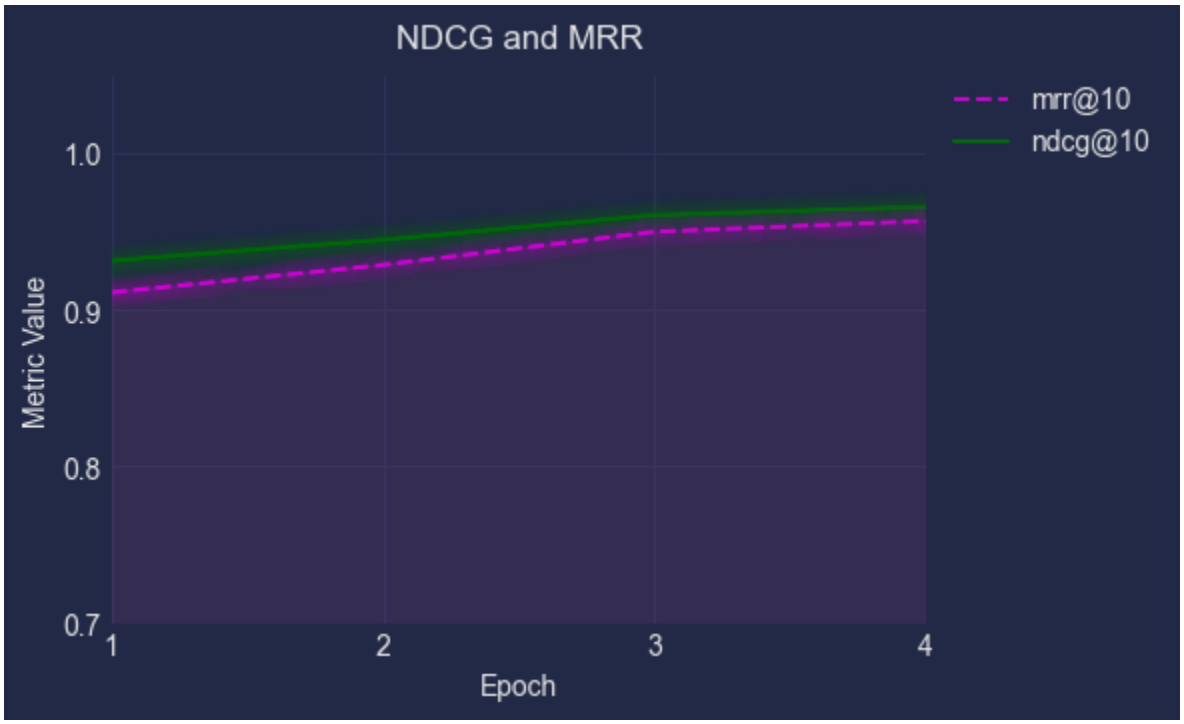
Accuracy Training History



Recall Training History



MRR & NDCG Training History



Model Evaluation and Comparison

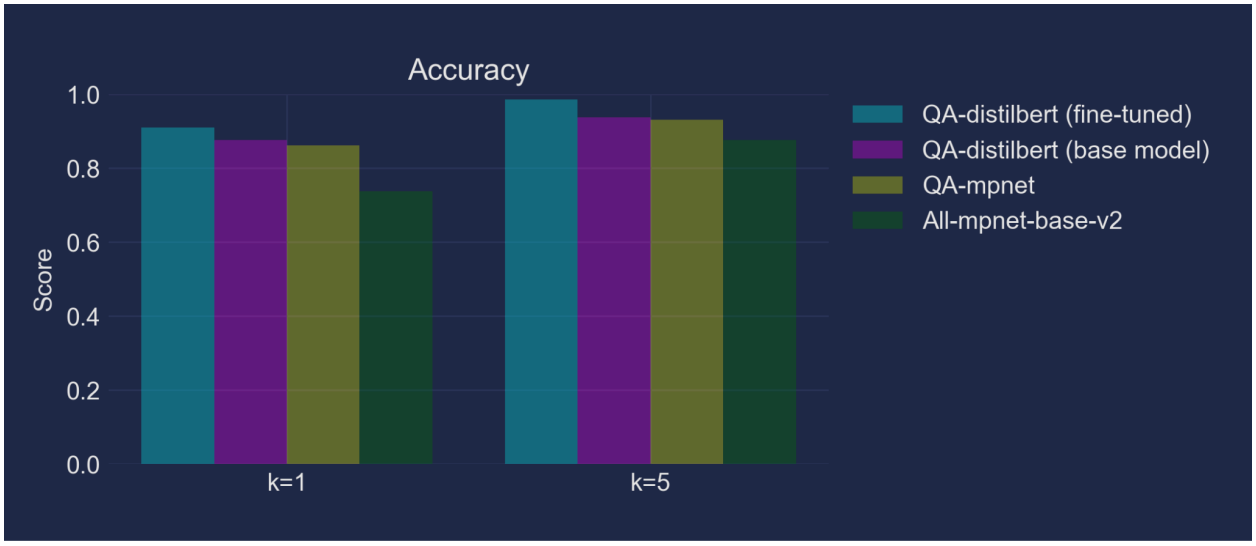
After training the final model with the best hyperparameters, its performance was evaluated using a holdout test set and compared against three other models: the pre-trained *multi-qa-distilbert-cos-v1*, *multi-qa-mpnet-base-dot-v1*, and *all-mpnet-base-v2*. The performance of the four models along various performance metrics are displayed below:

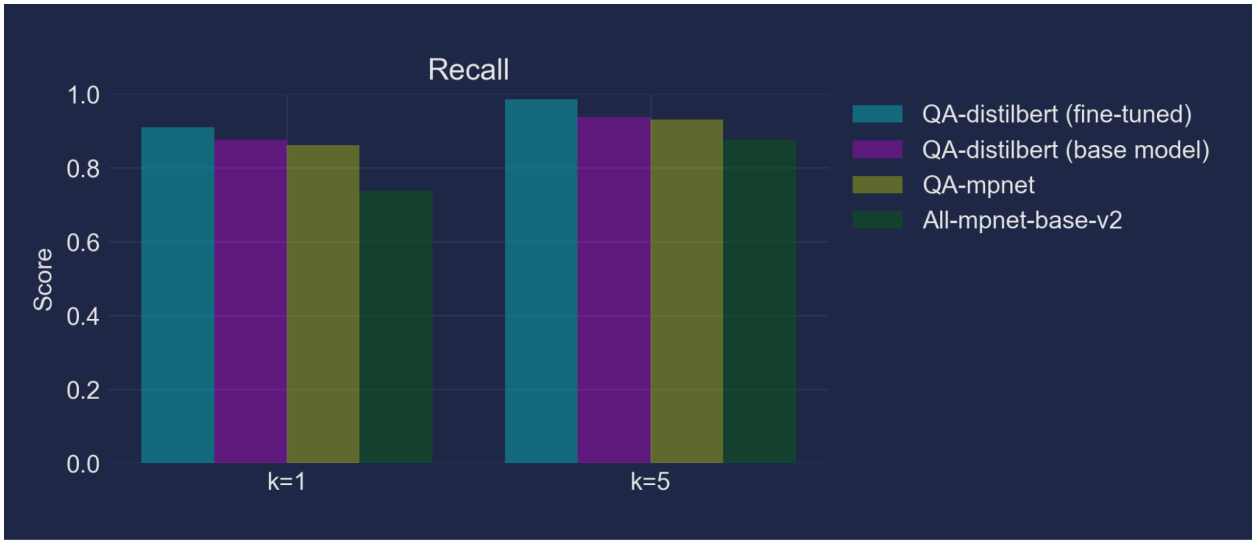
Table 1: Performance Metrics for Each Embedding Model

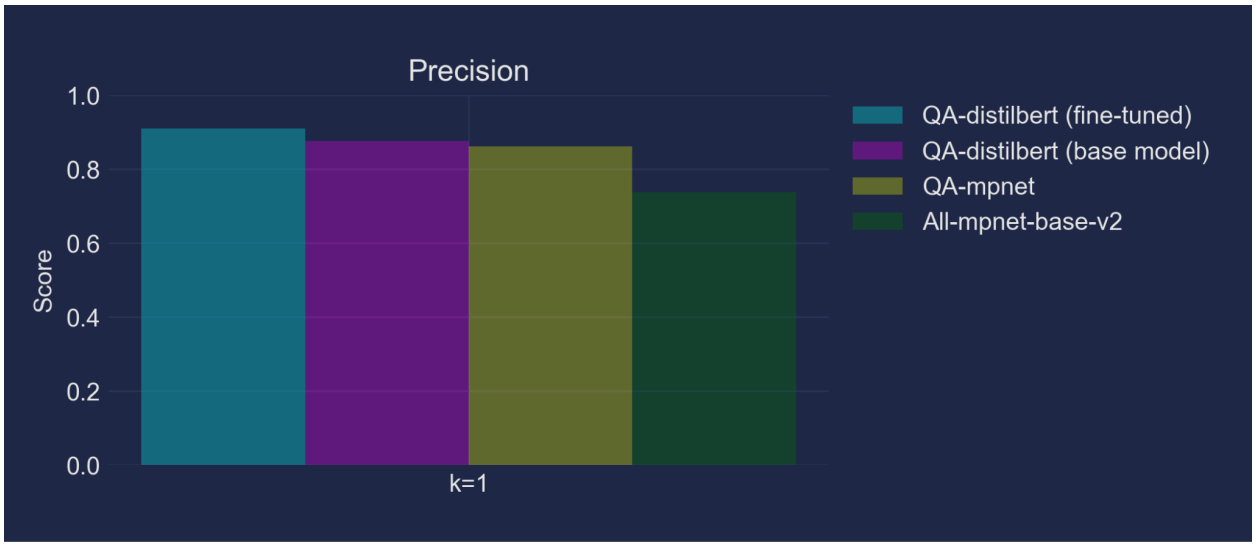
Metric	QA-distilbert (fine-tuned)	QA-distilbert (base model)	QA-mpnet	All-mpnet-base-v2
Accuracy@1	0.9103	0.8759	0.8621	0.7379
Accuracy@5	0.9862	0.9379	0.9310	0.8759
Precision@1	0.9103	0.8759	0.8621	0.7379
Recall@1	0.9103	0.8759	0.8621	0.7379
Recall@5	0.9862	0.9379	0.9310	0.8759
NDCG@10	0.9586	0.9142	0.9027	0.8276
MRR@10	0.9449	0.9020	0.8892	0.8028

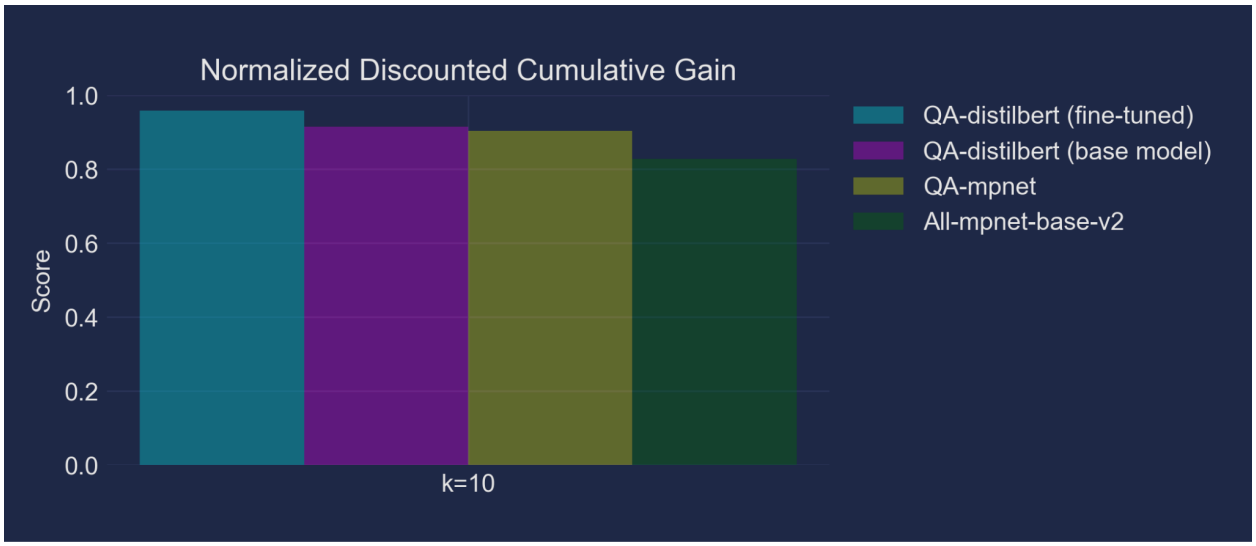
In all of the performance metrics, the fine-tuned DistilBERT model consistently scored the highest, followed by the pre-trained base model (multi-qa-distilbert-cos-v1), then the multi-qa-mpnet-base-dot-v1 model, and finally the all-mpnet-base-v2 model.

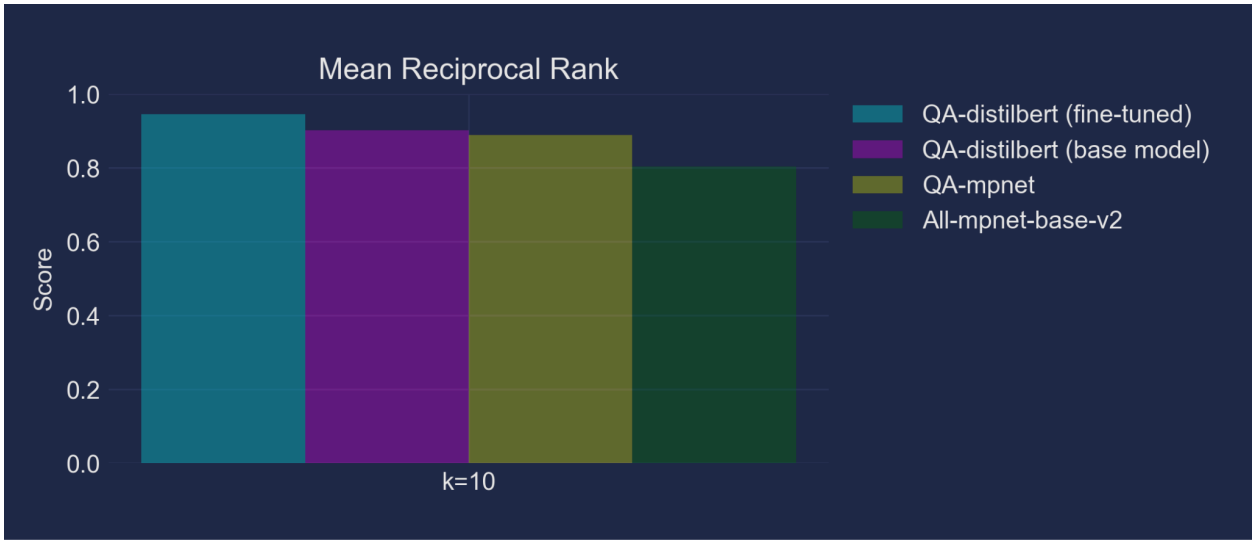
These results demonstrate that while the pre-trained base model (multi-qa-distilbert-cos-v1) performed well out-of-the-box, its performance was further improved by fine-tuning it on our company's dataset. The fine-tuning process allowed the model to adapt to the specific domain and characteristics of our data, leading to superior performance compared to the other models.




Recall
Score
1.0
0.8
0.6
0.4
0.2
0.0
k=1
k=5
QA-distilbert (fine-tuned)
QA-distilbert (base model)
QA-mpnet
All-mpnet-base-v2


Precision
Score
1.0
0.8
0.6
0.4
0.2
0.0
k=1
QA-distilbert (fine-tuned)
QA-distilbert (base model)
QA-mpnet
All-mpnet-base-v2


Normalized Discounted Cumulative Gain
Score
1.0
0.8
0.6
0.4
0.2
0.0
k=10
QA-distilbert (fine-tuned)
QA-distilbert (base model)
QA-mpnet
All-mpnet-base-v2



Conclusion

In this project, we successfully developed and fine-tuned a Question Answering Information Retrieval System to answer queries a customer may have for the company.

The data preparation phase involved converting the company's tabular data into human-readable text documents, enabling natural language processing techniques to be applied. A large language model was then employed to generate a diverse set of synthetic questions based on the information in the documents, creating a comprehensive training dataset.

Through our initial exploratory analysis, the *multi-qa-distilbert-cos-v1* model was identified as the most suitable pre-trained embedding model for our task. Bayesian optimization techniques were utilized to find the optimal hyperparameters that maximize its performance on our dataset. Ultimately, the fine-tuned model consistently outperformed other pre-trained models across various evaluation metrics, demonstrating the effectiveness of the fine-tuning process and the model's ability to adapt to the specific domain and characteristics of our data.

The significant performance improvements observed after fine-tuning the pre-trained model highlight the importance of tailoring the model to the specific domain and data characteristics. This approach can be extended to other domains and industries to enhance the effectiveness of question-answering systems. As the company's product catalog evolves, the system can be further refined by incorporating new data and retraining the model, enabling it to adapt to changes in the product offering and customer inquiries.

Limitations and Future Directions

A potential limitation of the current system is that the generated queries are relatively simplistic, requiring only a single relevant document to formulate an answer. In reality, more complex customer inquiries may necessitate retrieving and combining information from multiple documents to provide a comprehensive response. To address this, future work could explore expanding the QA system to retrieve multiple relevant documents and leverage large language models to synthesize and generate answers by consolidating information from various sources.

Another intriguing direction for future research is the integration of large language model chains with structured database queries. This approach could involve using language models to generate SQL code that directly queries various data tables within the company's database, retrieving the required information to formulate responses to customer inquiries. By combining the strengths of language models and structured data retrieval, this method could potentially provide more accurate and complete answers to complex questions, while also enabling dynamic updates as the underlying data evolves.

Overall, this project demonstrated the potential of leveraging natural language processing and machine learning techniques to enhance customer support and sales processes in the medical technology industry. By continuously improving and expanding the system's capabilities, it can contribute to better customer experiences, increased operational efficiency, and a competitive advantage for the company.