**Fine-Tuning a Large Language Model for Domain-Specific Tasks**

**Overview:**In this project, we have tuned a Large Language Model (LLM) to answer domain-specific questions regarding cancer-related data using a dataset named CancerQA.csv. The model was tuned with Retrieval-Augmented Generation (RAG) for better response generation. The procedure included several steps such as data preparation, tokenization, model tuning, and assessment. We have utilized Hugging Face Transformers, FAISS, Pinecone, and Streamlit in Google Colab to accomplish this project.

**Step 1: Preparing the Data**

The CancerQA.csv data was utilized in this task. The dataset includes columns such as category and resume with information of cancer-specific questions and answers. I started with loading and preprocessing the dataset, handling missing values, and removing unused columns so that the data was ready for training.

**Step 2: Data Splitting**

In order to facilitate effective model training and validation, the dataset was split into training and validation sets. An 80-20 split was used for this split to offer enough data to validate the performance of the model upon training.

**Step 3: Tokenization**The text data in the resume column was tokenized and converted to a numerical format that could be used as model input. Hugging Face's AutoTokenizer was used to tokenize the inputs, which were padded and truncated to accommodate the varying lengths of text.  
 **Step 4: Labeling**Because there were no explicit labels in the dataset, I labeled it using the category column. This allowed the model to be trained in a supervised environment, where it could learn to convert each cancer-related question into a suitable response from the resume column.

**Step 5: Develop Training Arguments and the Model**For fine-tuning, I used a pre-trained T5 model. I set the training parameters using Hugging Face's TrainingArguments class. The parameters included batch size, learning rate, number of epochs, and weight decay. Logging and checkpointing were enabled to keep track of training progress and save the model at regular intervals.   
 **Step 6: Train the Model with the Trainer Class.**The model was trained using the Hugging Face Trainer class, which handled the model's weight optimization from training data across multiple epochs. The training loss was monitored to ensure that the model learned properly from the data.

**Step 7: Error and Troubleshooting**Several issues arose during deployment, including incorrect column names and missing labels. These have been addressed by:   
  
Ensure that the correct column was used for tokenization.   
Applying the labels correctly for supervised training.   
To fix padding errors, set tokenizer.pad\_token = tokenizer.eos\_token.

**Step 8: Model Evaluation.**After training, I validated the model with the validation set. The model's performance was tracked using metrics such as accuracy and loss. These tests were used to fine-tune the model so that it could provide accurate answers to cancer-related questions.

**Conclusion:**I have optimized a T5 model to develop a Retrieval-Augmented Generation (RAG) model to answer questions related to cancer in this project. Through data preprocessing, tokenization, optimization of the model, and assessment, the model was optimized for performing tasks with a domain specialization. This optimization process was educative for large language model optimization for specialized functions. The final model is able to generate accurate, context-specific answers to cancer-related questions, which will enhance knowledge retrieval in medical scenarios.

**Screen Shots:**







