

Private GPT For Personalized Finance Management (PFM)

Introduction

Efficient data retrieval is essential in financial technology, where timely and accurate information can significantly impact decision-making. This project focuses on the integrating a large language model with Lang Chain to translate natural language queries into SQL commands, allowing users to interact seamlessly with complex databases. By harnessing advanced natural language processing capabilities, the system enhances user accessibility and improves the accuracy of financial data retrieval.

Objective

To develop an AI-powered chatbot, PrivateGPT, that integrates local and public datasets to provide personalized finance management services.

Experimental Setup

Platforms Used:

- **Google Colab:** A cloud-based platform providing free access to computational resources.
 - **Google Colab Free Tier Specifications:**
 - RAM: Approximately 12.68 GB
 - Disk: Approximately 67.2 GB
 - GPU: Intel Xeon CPU @2.20 GHz
 - GPU Memory: 15 GB GDDR5
 - Compute Capability: 7.5
- **VS Code:** An integrated development environment (IDE) used for code development and debugging.

Local Machine Specifications:

- **RAM:** 12 GB
- **Disk:** 512 GB SSD

Database Used:

- **MySQL Workbench:** An integrated tool for MySQL database administration and management, used for storing and managing the bank transaction data.

Methodology

1. Data Collection and Preparation

The objective was to acquire, clean, and preprocess data for subsequent analysis and model training. A sample dataset of 116,000 bank transactions in CSV format served as the data source. The CSV file underwent a cleaning and preprocessing stage to ensure data quality and consistency. The cleaned dataset was then loaded into MySQL Workbench for efficient data management.

2. Large Language Model (LLM) Setup

The Llama-2-7b ("meta-llama/Llama-2-7b-hf",[link](#)) model from Hugging Face was used for the project. The model is specifically designed for natural language understanding and generation as it excels in various task such as question answering, summarization and text generation. The model utilizes the capabilities of the Hugging Face Transformer library, allowing for seamless integration and fine tuning within the Hugging Face ecosystem.

3. Training Dataset Generation

To train the model, a custom dataset was created ([link](#)) which included SQL schema, instructions, and corresponding SQL queries. The Standard data format used for fine tuning the llama 2 model is of the following format.

```
<s>[INST] <<SYS>>
System prompt
<</SYS>>

User prompt [/INST] Model answer </s>
```

Considering this standard format, the custom data was also made accordance with the same format. Here is a sample of the data format .

```
<s>[INST] <<SYS>> You are an expert in converting natural
language questions into MYSQL queries. Here is the
database schema defined by create statement. CREATE TABLE
transactions ( `Account_No` VARCHAR(50) NOT NULL,
`Transaction_details` TEXT, `Withdrawal_amount` INTEGER,
`Deposit_amount` INTEGER, `Balance_amount` INTEGER,
`Value_date` DATE, `Date` DATE ) . Please write me a SQL
query that answers the given user question: <</SYS>> Give
me the breakdown of amount I spent each day of this
month? [/INST] SELECT DATE(Value_date) AS Date,
SUM(Withdrawal_amount) AS Daily_Spending FROM
transactions WHERE MONTH(Value_date) =
MONTH(CURRENT_DATE()) AND YEAR(Value_date) =
YEAR(CURRENT_DATE()) GROUP BY DATE(Value_date) </s>
```

3. Training the Model

Directly fine-tuning all parameters of large models (all 7 billion parameters in Llama-2) is costly and resource-intensive. Therefore, Parameter-Efficient Fine-Tuning (PEFT) strategies are used for optimal fine-tuning with fewer parameters. This technique involves freezing the pre-trained model weights (Llama-2 7B) and fine-tuning with a smaller set of parameters.

Low-Rank Adaptation (LoRA) is a novel technique for fine-tuning large language models (LLMs) that significantly reduces the number of trainable parameters while maintaining or even improving their performance. This is achieved by injecting smaller, trainable matrices into each layer of the LLM's architecture, rather than directly modifying the original weights.

```
# Set supervised fine-tuning parameters
trainer = SFTTrainer(
    model=model,
    train_dataset=dataset['train'],
    eval_dataset=dataset["test"],
    peft_config=peft_config,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    tokenizer=tokenizer,
    args=training_arguments,
    packing=packing,
)
```

The model was trained on two different dataset. The first dataset consists of custom created user natural language queries related to finance and their corresponding SQL queries. This dataset was crucial for helping the model accurately respond to user queries, thereby minimizing LLM hallucinations.

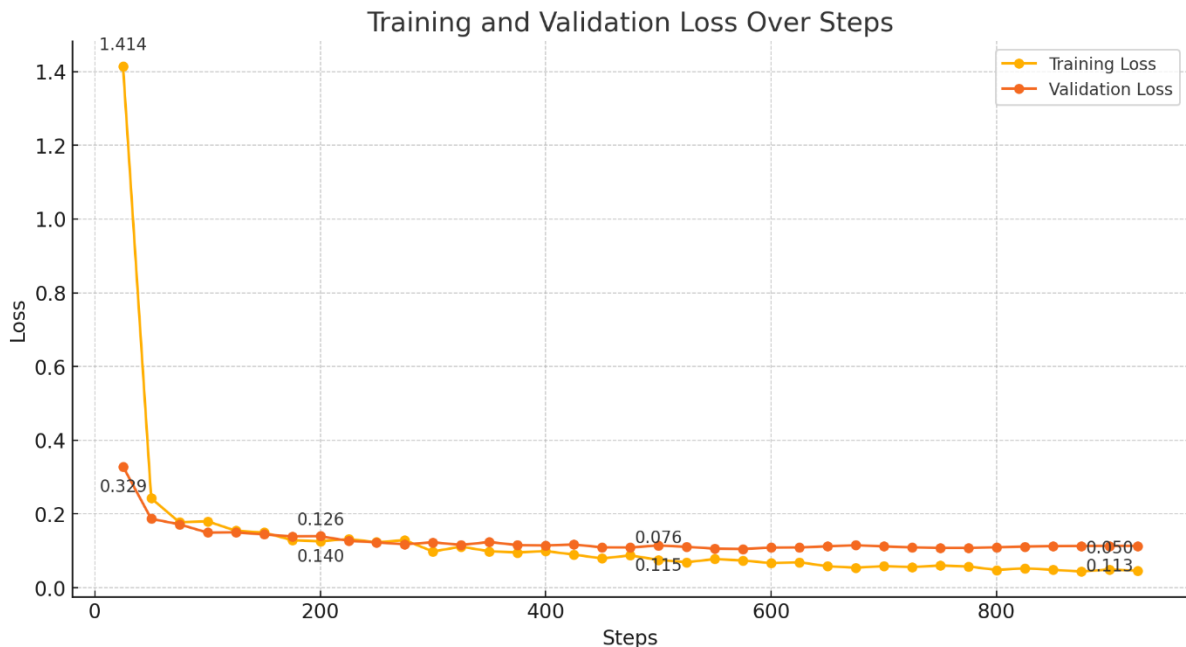
Since the data consisted of a relatively small number of data (around 350), determining the optimal number of epochs for training the model was challenging. Higher Epoch could cause the LLM model to memorize the data rather than learn patterns, while a lower epoch might not be enough for model to identify the relationships and patterns in data. So training was conducted multiple times with varying hyper parameters such and epoch, batch size, learning rate ,learning scheduler, weight decay, rank ,scaling factor etc to identify the best result for the LLM .

Ultimately, following configuration was chosen to train the model because this configuration performed better compared to other epoch settings. You can view the experiments related to varying epoch configurations on the [HuggingFace Account](#) .

The Configuration:

- Learning Rate: 2e-4
- Epochs: 6
- Batch Size: 2
- Rank: 64

- Scaling Factor (lora_alpha): 16
- Optimizer: paged_adamw_32bit
- Weight Decay: 0.001



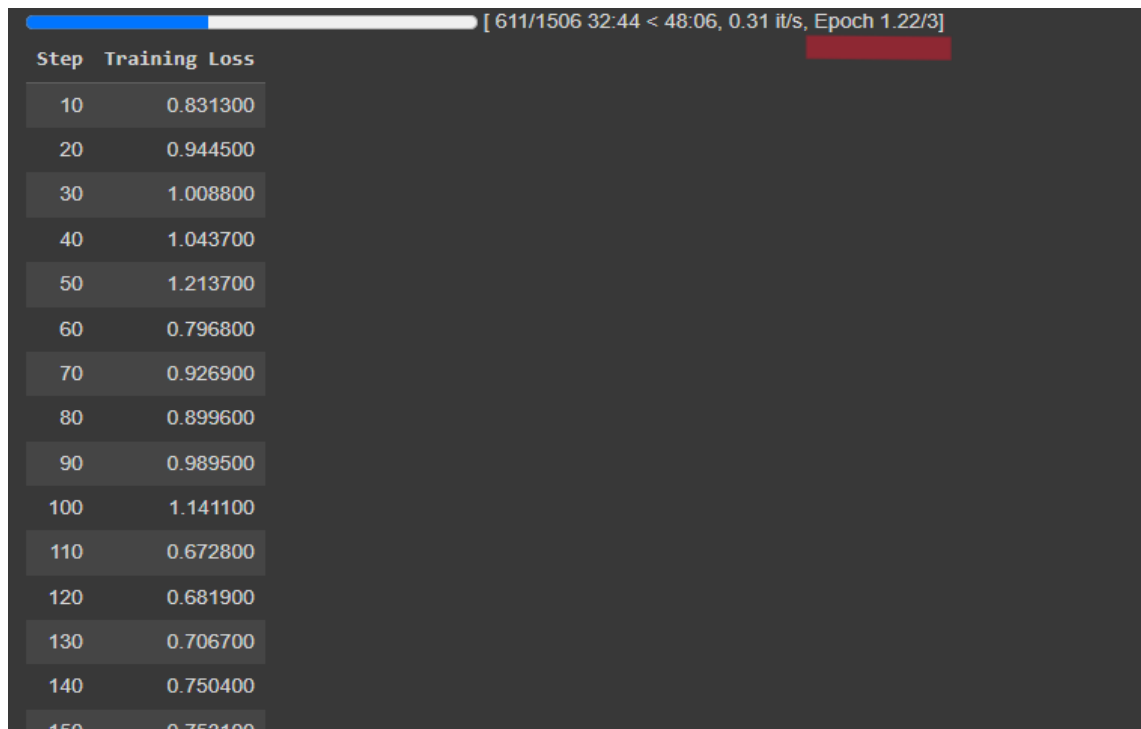
The above diagram illustrates the training and validation loss during model training. Overfitting occurs when a model learns the training data too well, capturing noise and details that do not generalize to new, unseen data. This typically results in a significant gap between training loss (which decreases) and validation loss (which increases). In contrast, the diagram shows no signs of overfitting, as the training loss and validation loss remain close to each other throughout the process, with no significant fluctuations in the validation loss. This suggests that the model generalizes well to new data and has not excessively memorized the training examples.

The second dataset, which included 1,500 records, focused on reasoning ([link](#)), designed to help the LLM break down complex questions into smaller, manageable tasks. This approach aims to enhance the model's ability to provide precise and contextually appropriate answers, reducing errors and improving overall performance. The data format was as follows:

```
<s> [INST] I want to buy a car worth Rs. 5,00,000 within 2 years. How much should I save each month if my current average monthly saving is Rs. 9,385.33?[/INST] To determine how much you should save each month, follow these steps: 1. Determine Total Number of Months: The time frame is 2 years. Since there are 12 months in a year, the total number of months is: 2 years × 12 months/year = 24 months 2. Calculate Required Monthly Savings: The target amount to save is Rs. 5,00,000. The required monthly savings is calculated by dividing the total amount by the total number of months: Rs. 5,00,000 / 24 months = Rs. 20,833.33 per month 3. Compare Current Savings with Required Savings: Your current average monthly savings is Rs. 9,385.33. The additional amount needed to save each month is: Rs. 20,833.33 - Rs. 9,385.33 = Rs. 11,448 4. Conclusion: To reach the goal of saving Rs. 5,00,000 in 2 years, you need to increase your monthly savings to Rs. 20,833.33. The additional amount you need to save each month is Rs. 11,448 </s>
```

The data format included detailed problem statements, intermediate reasoning steps, and final answers, allowing the model to learn not only the solution but also the process of arriving at it. This comprehensive format supports the development of a more robust and nuanced understanding of complex queries. Each record was crafted to ensure that the model could understand and replicate the reasoning process, that helps in improving its capability to handle

similar complex questions in real-world scenarios. This structured approach allows for a better grasp of logical sequences and decision-making patterns.



A terminal window showing the progress of a training process. At the top, a progress bar is partially filled with blue, followed by a status string: "[611/1506 32:44 < 48:06, 0.31 it/s, Epoch 1.22/3]". Below this, a table displays the training loss for each step from 10 to 150. The table has two columns: "Step" and "Training Loss".

Step	Training Loss
10	0.831300
20	0.944500
30	1.008800
40	1.043700
50	1.213700
60	0.796800
70	0.926900
80	0.899600
90	0.989500
100	1.141100
110	0.672800
120	0.681900
130	0.706700
140	0.750400
150	0.753100

4. Merging the Trained model with Base model

After fine tuning using the PEFT technique, the next step involved merging the fine tuned model with the base model. The result is a unified model that preserves the general knowledge of the base model while incorporating specialized improvements. This approach ensures that the final model benefits from both the broad capabilities of the base model and the specific advancements obtained through targeted fine-tuning.

```
# Reload model in FP16 and merge it with LoRA weights
base_model = AutoModelForCausalLM.from_pretrained(
    model_name,
    low_cpu_mem_usage=True,
    return_dict=True,
    torch_dtype=torch.float16,
    device_map=device_map,
)
model = PeftModel.from_pretrained(base_model, new_model)
model = model.merge_and_unload()
```

5. Model Conversion and Storage

The trained LLM model was not directly suitable for inference due to its original format. Therefore, it was quantized into 5-bit(q5_0) GGUF (GPT-Generated Unified Format), which is specifically designed for efficient inference on low computational devices. Subsequently, the model was stored in the Hugging Face repository, making it available for local download and further use. ([link](#))

```
QUANTIZATION_METHODS = ["q5_0"]

for method in QUANTIZATION_METHODS:
    qtype = f"{MODEL_NAME}/{MODEL_NAME.lower()}.{method.upper()}.gguf"
    !./quantize {fp16} {qtype} {method}s
```

6. Database Connection and RAG(Retrieval Augmented Generation)

A connection was established with MySQL Workbench, where the actual transaction data was stored. During the implementation, a few-shot prompt template was created as part of the RAG (Retrieval-Augmented Generation) Architecture. This template included a diverse range of finance-related questions that users might typically ask, as well as examples of natural language questions where the LLM had previously failed to generate accurate responses. Each entry in the few-shot template comprised the user's question, the corresponding SQL query, the result from MySQL Workbench, and the accurate answer. This comprehensive few-shot template served as a reference for the LLM when generating SQL queries for new, unseen natural language questions. By including both previously failed queries and a broad spectrum of finance-related queries, the template provided a robust foundation for improving the LLM's response accuracy.

```
{
  "Question": "What is my average spendings of last 11 transactions?",
  "SQLQuery": """"SELECT AVG(Withdrawal_amount) AS Average_Spending FROM (SELECT Withdrawal_amount FROM transactions WHERE Withdrawal_amount
  IS NOT NULL ORDER BY Value_date DESC LIMIT 11) AS Last11Transactions""",
  "SQLResult": "1138377.5455",
  "Answer": "1138377.5455 is your average spendings of last 11 transactions ."
},
{
  "Question": "How much did I spend each quarter this year?",
  "SQLQuery": """"SELECT QUARTER(Value_date) AS Quarter, SUM(Withdrawal_amount) AS Total_Spending
  FROM transactions WHERE YEAR(Value_date) = YEAR(CURRENT_DATE()) GROUP BY QUARTER(Value_date)""",
  "SQLResult": """"[(1, Decimal('5885524375')),
  (2, Decimal('2689194690')),
  (3, Decimal('263256582'))]""",
  "Answer": """"Spendings for each quater is :
  Quarter-1:RS. 5885524375, Quarter-2:RS. 2689194690, Quarter-3:RS. 263256582""""
},
}
```

7. Embedding and Vector Store Creation

The few-shot template was embedded using the "all-mpnet-base-v2" sentence transformer from Hugging Face, which excels in semantic search compared to other embeddings. The resulting embeddings were then stored in the Chroma DB vector store. When a new user question is posed, the system performs a semantic search to identify the top 2 most relevant questions from the vector store. For each of these top 2 questions, the corresponding SQL query, result, and answer are retrieved. The LLM then generates a response for the new user's unseen question by referencing these top 2 similar questions, their SQL queries, results, and answers, thereby improving the accuracy and relevance of the generated response.

ST model name	Sentence embedding Performance (14 semantic similarity tasks)	Model size [#parameters]
all-mpnet-base-v1	69.98	110M
all-mpnet-base-v2	69.57	110M
paraphrase-mpnet-base-v2	67.97	110M

8. Additional Information and Guard Rails Implementation

The LLM was provided with additional necessary information about the database schema to ensure that the model does not use unnecessary columns that were not present in MySQL. User questions (in natural language) were taken as input, and necessary guard rails were implemented to ensure the relevance and safety of the queries. These guard rails ensured that:

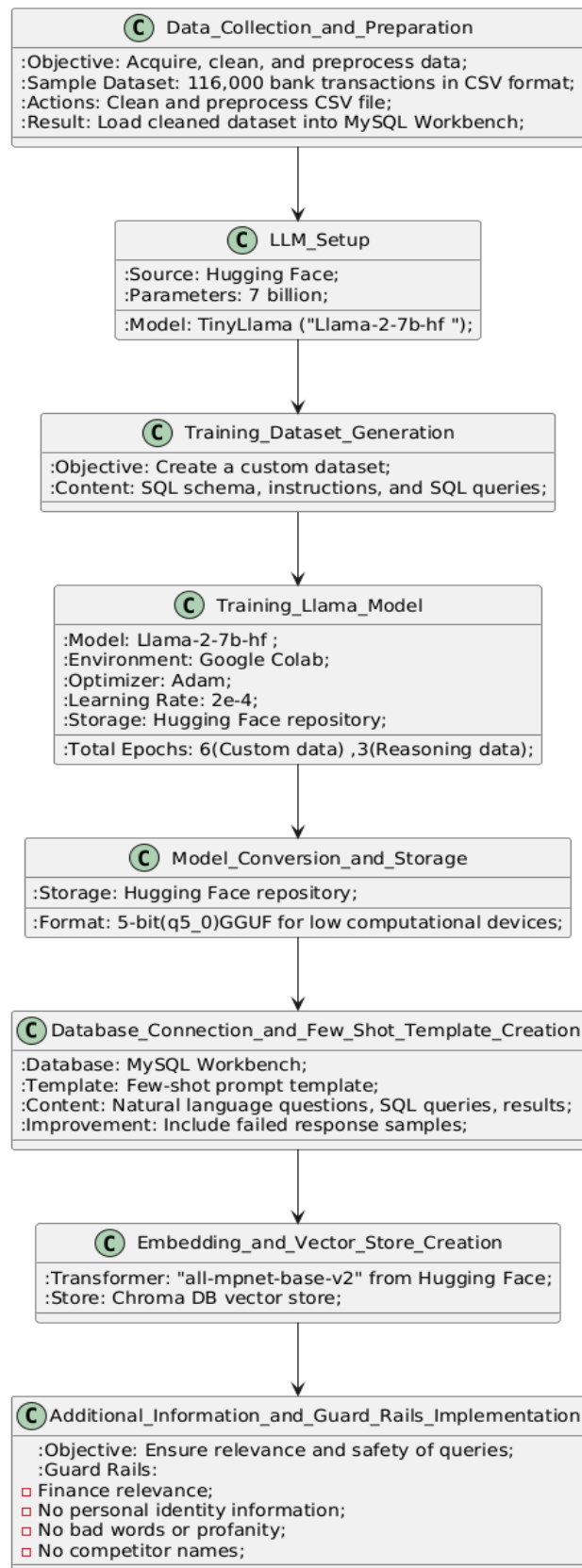
- User questions were relevant to finance.
- User questions did not contain any personal identity information such as bank account numbers.
- User questions did not contain any bad words, toxic language, or profanity.
- User questions did not involve competitor companies names .

```
guard = Guard().use_many(  
    CompetitorCheck(["khalti", "fusemachine", "Deerhold", "Deerwalk", "Cotiviti"], on_fail=OnFailAction.EXCEPTION),  
    ToxicLanguage(threshold=0.9, validation_method="sentence", on_fail=OnFailAction.EXCEPTION),  
    ProfanityFree(on_fail=OnFailAction.EXCEPTION)
```

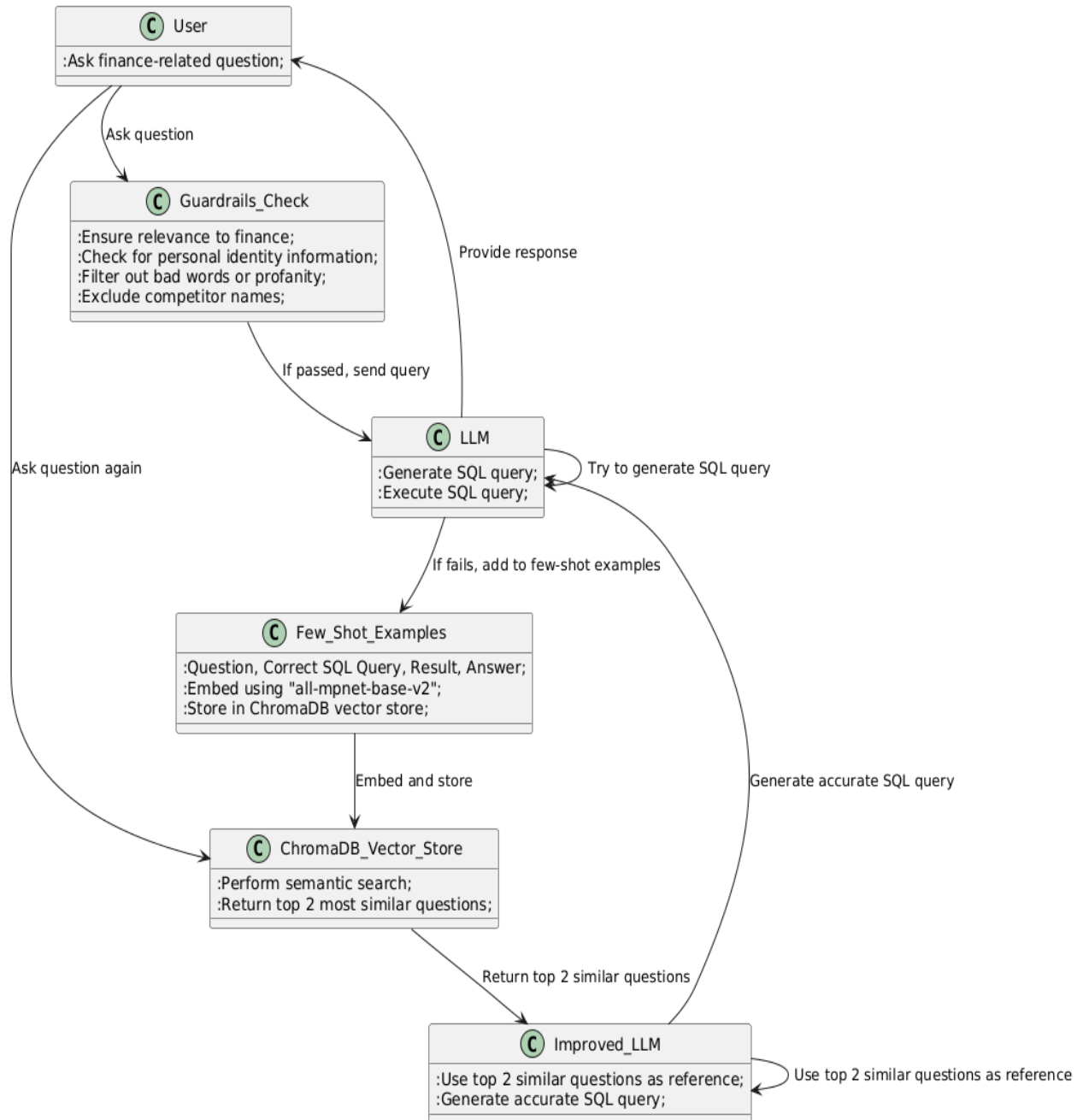
If the user question passed the guard rails, it was given to the LLM to generate the SQL query, execute the corresponding query in MySQL Workbench to retrieve the relevant data, and respond with the accurate answer.

Project Implementation Flow Diagrams

Full Project Implementation Flow for LLM Training, RAG Implementation, Response Generation



Detailed Workflow for LLM Response Generation Using RAG Architecture



Results

The response of the LLM model in generating SQL queries from natural language inputs was satisfactory, effectively retrieving relevant data from the MySQL database.

1. Possible Descriptive Statistics to deliver :

Question 1: How much I earned in total last month?

```
> Entering new SQLiteDatabaseChain chain...
How much I earned in total last month ?as my account number is 409000611074.
SQLQuery:SELECT SUM(Deposit_amount) AS Total_Income
FROM transactions
WHERE YEAR(Value_date) = YEAR(CURDATE())
AND MONTH(Value_date) = MONTH(CURDATE()) - 1
AND Account_No = '409000611074';
SQLResult: [(Decimal('5900000'),)]
Answer:You earned RS. 5900000 Last month .
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 2: How many fund transfer I did in last 30 days?

```
> Entering new SQLiteDatabaseChain chain...
How many fund transfer I did in last 30 days? as my account number is 409000438611
SQLQuery:SELECT COUNT(*) AS Total_Fund_Transfers FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 30 DAY) AND ACCOUNT_NO= '409000438611'
SQLResult: [(285,)]
Answer:You did 285 fund transfer in Last 30 days.
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 3: What is my consistent income in my last 3 months?

```
> Entering new SQLiteDatabaseChain chain...
What is my consistent income in my last 3 months ?
SQLQuery:SELECT AVG(Deposit_amount) AS Average_Income FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH)
SQLResult: [(Decimal('247265.3232'),)]
Answer:My consistent income in my last 3 months is:
3 months ago: 247265.3232
```

Status: Failed to generate correct SQL Query .The SQL query generated by llm calculates the average deposit amount across all transactions in the last 3 months, not the average monthly income (Assuming the "consistent income" refers to some measure of stability or average income over a period).

So we added the Question along with its corresponding correct SQL query, SQL Result and proper format of response in few-shot learning template.

```
{
  'Question': "What is my consistent income in last 2 months as my account number is 409000493201 ?",
  'SQLQuery': ""SELECT AVG(Monthly_Income) AS Consistent_Income
FROM (SELECT YEAR(Value_date) AS Year, MONTH(Value_date) AS Month, SUM(Deposit_amount) AS Monthly_Income FROM transactions WHERE Value_date >=
DATE_SUB(CURRENT_DATE(), INTERVAL 2 MONTH) AND ACCOUNT_NO= '409000493201'
GROUP BY YEAR(Value_date), MONTH(Value_date)) AS Income_Last_3_Months;
""",
  'SQLResult': "2466666.6667",
  'Answer': "The consistent income of last 3 months is RS. 2466666.6667"
```

Next time the same question was asked, the model was able to respond accurately.

```
> Entering new SQLiteDatabaseChain chain...
What is my consistent income in my last 3 months? as my account number is 409000438611
SQLQuery:SELECT AVG(Monthly_Income) AS Consistent_Income FROM (SELECT YEAR(Value_date) AS Year, MONTH(Value_date) AS Month, SUM(Deposit_amount) AS Monthly_Income FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH) AND ACCOUNT_NO= '409000438611'
GROUP BY YEAR(Value_date), MONTH(Value_date)) AS Income_Last_3_Months;
SQLResult: [(Decimal('16593987.7500'))]
Answer:The consistent income of last 3 months is $16593987.7500
> Finished chain.
```

Question 4: What are my total expenses for the last three months ?

```
> Entering new SQLiteDatabaseChain chain...
What are my total expenses for the last three months? as my account number is 409000611074
SQLQuery:SELECT SUM(Withdrawal_amount) AS Total_Expenses FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 3 MONTH) AND ACCOUNT_NO= '409000611074'
SQLResult: [(Decimal('20786482'))]
Answer:You have total expenses of Rs. 20786482 for Last three months.
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 5: What was my total spending last week?

```
> Entering new SQLiteDatabaseChain chain...
What was my total spending last week? as my account number is 409000438611
SQLQuery:SELECT year(Value_date) as Year,sum(Withdrawal_amount) AS Total_Expenses FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 WEEK) AND ACCOUNT_NO= '409000438611' group by year(Value_date)
SQLResult: [(2024, Decimal('35601373'))]
Answer:You spent following amount each day last week:
Year: 2024
Month: 6
Day: 1373
```

Status: Successful query generation, execution, fetching, but there was Error in sentence formation while responding.

So we added the Question along with its corresponding correct SQL query, SQL Result and proper format of response in few-shot learning template.

```
{
  'Question': "What was my total spending last week? as my account number is 409000438611 ",
  'SQLQuery': ""SELECT sum(Withdrawal_amount) AS Total_Expenses FROM transactions WHERE YEAR(Value_date) = YEAR(CURDATE()) AND week(Value_date) = week(CURDATE())-1
AND ACCOUNT_NO= '409000438611'""",
  'SQLResult': "33201373",
  'Answer': "Your total expenses last week was RS. 33201373."
```

Next time the same question was asked to the model and it was able to respond accurately.

```
> Entering new SQLiteDatabaseChain chain...
What was my total spending last week? as my account number is 409000438611
SQLQuery:SELECT SUM(Withdrawal_amount) AS Total_Spending FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 WEEK) AND ACCOUNT_NO= '409000438611'
SQLResult: [(Decimal('35601373'))]
Answer:You spent total RS. 35601373 Last week.
I hope this helps you!
```

Question 6: Can you give me a breakdown of my expenses for each day this week?

```
> Entering new SQLiteDatabaseChain chain...
Can you give me a breakdown of my expenses for each day this week? as my account number is 409000493201
SQLQuery:SELECT DATE(Value_date) AS Date, SUM(Withdrawal_amount) AS Total_Expenses FROM transactions WHERE YEAR(Value_date) = YEAR(CURDATE()) and week(Value_date) = week(CURDATE()) and account_no='409000493201' GROUP BY DATE(Value_date)
SQLResult: [(datetime.date(2024, 7, 21), Decimal('62112')), (datetime.date(2024, 7, 22), Decimal('10331'))]
Answer: You spent following amount each day this week:
        July 21:DS. 62112, July 22:DS. 10331
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation. (Note : Instead of Rupees (RS) it displayed DS).

Question 7: How many transactions did I make last month?

```
> Entering new SQLiteDatabaseChain chain...
How many transactions did i make last month as my account number is 409000611074
SQLQuery:SELECT COUNT(*) AS Total_Transactions FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL 1 MONTH) AND ACCOUNT_NO= '409000611074'
SQLResult: [(1,)]
Answer: You made 1 transaction Last month.
> Finished chain.
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 8: What was my total income last month?

```
> Entering new SQLiteDatabaseChain chain...
What was my total income last month? as my account number is 409000493201
SQLQuery:SELECT SUM(Deposit_amount) AS Total_Income FROM transactions WHERE YEAR(value_date) = YEAR(CURRENT_DATE()) AND MONTH(value_date) = MONTH(CURRENT_DATE()) - 1 AND Account_No = '409000425051' ;
SQLResult: [(Decimal('0'),)]
Answer: Your total income last month is (decimal 0) .
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation. (Note : Amount is 0 since there was no income last month for the user).

Question 9: How much did I save last month?

```
> Entering new SQLiteDatabaseChain chain...
How much did i save last month as my account number is 409000493201
SQLQuery:SELECT sum(Deposit_amount) - sum(Withdrawal_amount) AS TotalSavings FROM transactions WHERE Value_date >= DATE_SUB(CURRENT_DATE(), INTERVAL MONTH(409000493201) year)-1 AND ACCOUNT_NO= '409000493201'
SQLResult: [(None,)]
Answer: Savings of last month are: -856.75
> Finished chain.
```

Status: Error in Generation of correct SQL query.

So we added the Question along with its corresponding correct SQL query, SQL Result and proper format of response in few-shot learning template .

```
{
  'Question': "How much did i save last month as my account number is 409000493201 ",
  'SQLQuery': "" SELECT SUM(Deposit_amount) - SUM(Withdrawal_amount) AS Total_Savings
FROM transactions WHERE year(value_date) = year(curdate()) and month(value_date)=month(curdate()) -1 AND Account_No = '409000493201';
""",
  'SQLResult': "89804",
  'Answer': "Your savings last month was RS. 89804."
}
```

Next time the same question was asked to the model and it was able to respond accurately.

```
> Entering new SQLiteDatabaseChain chain...
How much did i save last month ? as my account number is 409000611074
SQLQuery:SELECT SUM(Deposit_amount) - SUM(Withdrawal_amount) AS Total_Savings FROM transactions WHERE YEAR(value_date) = YEAR(CURRENT_DATE()) AND MONTH(value_date)=MONTH(CURRENT_DATE()) -1 AND ACCOUNT_NO= '409000611074';
SQLResult: [(Decimal('-1314534'),)]
Answer:Your savings last month was RS. -1314534.
```

2. Personal Finance Management

Question 2.1: Show me a summary of my spending for the past month.

```
> Entering new SQLiteDatabaseChain chain...
can you show me a summary of my spending for the past month ?
SQLQuery:SELECT COUNT(*) AS number_of_transactions,SUM(Withdrawal_amount) AS total_spending FROM transactions WHERE year(Value_date) =year(curdate()) And month(value_date) = (month(curdate()) -1);
SQLResult: [(2772, Decimal('529053340'))]
Answer:You made 2772 transaction , spend RS 529,053,340 in total . </s>
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 2.2: What is my current Net worth?

```
> Entering new SQLiteDatabaseChain chain...
What is my current net worth for account number 409000493201 ?
SQLQuery:SELECT Balance_amount AS Net_Worth FROM transactions where account_no='409000493201' ORDER BY Value_date DESC LIMIT 1;
SQLResult: [(743583,)]
Answer:Your current network is RS 74,35,83
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

Question 3.1: I want to save RS 120,000 for a vacation within a year. Can you help me set a goal ?

```
> Entering new SQLiteDatabaseChain chain...
I want to save RS 120,000 for a vacation within one year. Can you help me set a goal for account number 409000493201 ?
SQLQuery:
c:\Users\Prabal Kuinkel\Desktop\VARL-Intern\privatgpt\privatgpt\Lib\site-packages\langchain_core_api_deprecation.py:119: LangChainDeprecationWarning: The method "Chain.__call__" was deprecated in langchain 0.1.0 and will be removed in a future version.
warn_deprecated(
SELECT round(AVG(Monthly_Savings),0) AS Average_Monthly_Savings FROM (SELECT YEAR(Value_date) AS Year, MONTH(Value_date) AS Month, SUM(Deposit_amount - Withdrawal_amount) AS Monthly_Savings FROM transactions WHERE Account_No=409000493201)
SQLResult: [[{"decimal": "9083"}]]
Answer: So your average monthly saving is RS 9083 .
Let's think step by step:
step-1) Determine Total Number of Months:
Time frame: 1 year (12 months in 1 year So 1 year * 12 months = 12) = 12 months
step-2) Calculate Required Monthly Savings:
Target amount: RS 120,000
Required monthly savings: (It is obtained by dividing the total amount by Time frame) i.e RS. 120,000 / 12 months = RS. 10,000
step-3) Compare Current Savings with Required Savings:
Current average monthly savings: RS 9083
Difference needed: RS 10,000 - RS 9083 = RS. 8917
step-4) Conclusion: To reach the goal of saving RS 120,000 in 1 year, increase monthly savings to RS 10,000 .The additional amount to save each month is RS. 8917 </s>
```

Status: Successful query generation, execution, fetching, but calculation mistake since 10,000 – 9083 is 917 (as model responded 8917)

Question 3.2 Where Should I invest my extra savings ?

```
res-llm.invoke("Where should I invest my extra savings ? Explain in small points and format in proper order ? ")
print(res)
✓ 1m 37.8s

[/INST] Here are some options for investing your extra savings:
- High Yield Savings Accounts: These accounts typically offer higher interest rates than traditional savings accounts, but may have restrictions on withdrawals.
- Money Market Funds: These funds pool money from numerous investors and use it to purchase low-risk securities such as commercial paper, treasury bills, and certificates of deposit (CDs).
- Stocks: These represent ownership in a company and offer the potential for long-term growth. However, they can be volatile and carry more risk than other options.
- Bonds: These represent debt owed by a corporation or government and pay regular interest payments. They are generally less risky than stocks but may have lower returns.
- Real Estate Investment Trusts (REITs): These allow individuals to invest in real estate without directly owning physical
```

Status: Satisfactory response

Question 6.2 How can I save more money ?

```
res-llm.invoke("How can I save more money.Explain in points and format in proper order ? ")
print(res)
✓ 1m 32.5s

Step1: Set a goal, track expenses, and create a budget.
Step2: Cut back on unnecessary spending, such as eating out or buying expensive clothing.
Step3: Increase income by finding side hustles, asking for raises at work, or picking up additional gigs.
Step4: Use the 50/30/20 rule to allocate your budget - 50% towards necessities, 30% towards discretionary spending, and 20% towards saving and debt repayment.
Step5: Consider opening a separate savings account for long-term goals, such as retirement or buying a house.
Step6: Automate your savings by setting up automatic transfers from your checking to your savings account each month.
Step7: Review and adjust your budget regularly to make sure you're on track with your financial goals.
```

Status: Satisfactory response

Question 8.1: I want to buy a bike that cost RS 4,00,000 next year. How should I plan for it ?

```
> Entering new SQLDatabaseChain chain...
I want to buy a bike that cost RS 4,00,000 after 1 year. So how much amount should i save each month for account number 409000493201?
SQLQuery:SELECT round(AVG(Monthly_Savings),0) AS Average_Monthly_Savings
FROM (
  SELECT YEAR(Value_date) AS Year, MONTH(Value_date) AS Month,
        SUM(Deposit_amount - Withdrawal_amount) AS Monthly_Savings
  FROM transactions where Account_No='409000493201'
  GROUP BY YEAR(Value_date), MONTH(Value_date)
) AS Monthly_Savings_Calculation;
SQLResult: [(Decimal('9083'),)]
Answer: So your average monthly saving is RS 9083 .
  Lets think step by step:
    step-1) Determine Total Number of Months:
      Time frame: 1 year (12 months in 1 year So 1 year = 12 months) = 12 months
    step-2) Calculate Required Monthly Savings:
      Target amount: RS 4,00,000
      Required monthly savings:(It is obtained by dividing the total amount by Time frame) i.e RS. 4,00,000 / 12 months = RS. 33,333.33
    step-3) Compare Current Savings with Required Savings:
      Current average monthly savings: RS 9083
      Difference needed: RS 33,333.33 - RS 9083 = RS. 24,241.00
    step-4) Conclusion: To reach the goal of saving RS 4,00,000 in 1 year. Increase monthly savings to RS 33,333.33 .The additional amount to save each month is RS. 24,241.00
```

Status: Successful query generation, execution, fetching, and response with correct sentence formation.

3. Test for Guard Rails Implementation

Question 1: Does cat bark like dog?

Your Question is not Relevant to the Context

Status: Since the question asked by user was irrelevant, the question was discarded.

Question 2: Idiot what is my saving of this month?

Validation failed for field with errors: The following sentences in your response were found to be toxic:

- Idiot what is my saving of this month ?

Status: Since the question asked by user contains toxicity, the question was discarded.

Question 3: Shit What was my total income last month?

Validation failed for field with errors: The following sentences in your response were found to be toxic:

- shit What was my total income last month?

Status: Since the question asked by user contains toxicity, the question was discarded.

Question 4: Hey tell me detailed description of fusemachine.

Validation failed for field with errors: Found the following competitors: [['fusemachine']]. Please avoid naming those competitors next time

Status: Since the question asked by user contains information about other companies, the question was discarded.

Evaluation metrics

<div>C</div>	Confusion Matrix for LLM Response Evaluation

<div>C</div> C		
ConfusionMatrix	Accurate Response	Inaccurate Response
Correct SQL Query	TP (True Positive): 11	TN (True Negative): 2
Incorrect SQL Query	FP (False Positive): 0	FN (False Negative): 2

Explanation:

- 1. TP (True Positives): Correct SQL query generation with an accurate response (6 instances).
- 2. TN (True Negatives): Correct SQL query generation with an inaccurate response (1 instance).
- 3. FN (False Negatives): Incorrect SQL query generation with an inaccurate response (2 instances).

Conclusion

The project successfully demonstrated the integration of a large language model with Lang Chain to facilitate efficient data retrieval from natural language inputs, enabling intuitive access to financial data. The system effectively bridged the gap between technical complexity and user accessibility, showcasing the potential for real-world applications in the finance sector.

While the model performed satisfactorily, there remains room for improvement in handling complex queries and optimizing response times. Future work could focus on refining these areas to enhance user experience further.

Overall, this project sets the foundation for developing intelligent financial data retrieval systems that empower users to make informed decisions seamlessly.

References

1. Guardrails AI. (n.d.). *Hub - Guardrails AI*. from <https://hub.guardrailsai.com/>
2. Substratus AI. (2023, July 15). *Converting Hugging Face models to GGUF models*. from <https://www.substratus.ai/blog/converting-hf-model-gguf-model/>