Chat Bot using LLM.

Task

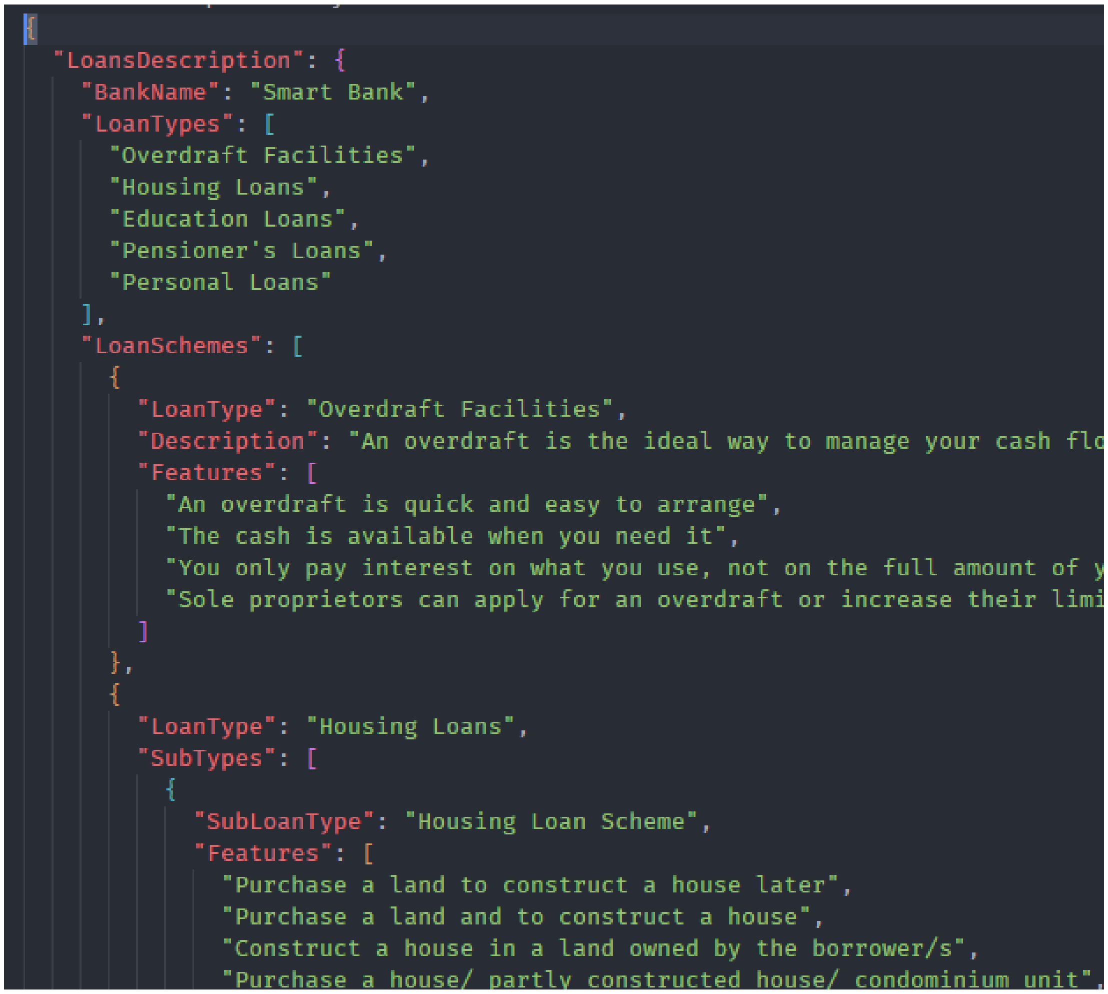
To develop a chat bot for a bank to aid their customers in inquiries about loans. It is required to use an LLM to complete this task.

Approach

Data Preparation

- The raw data provided included text information relevant to the loan scheme of the bank.

- It was required to break down the given data into chunks to be vectorized, but the data was not in a proper format so it impossible to extract data using a python script etc. so the data was converted into a proper JSON format manually as below.



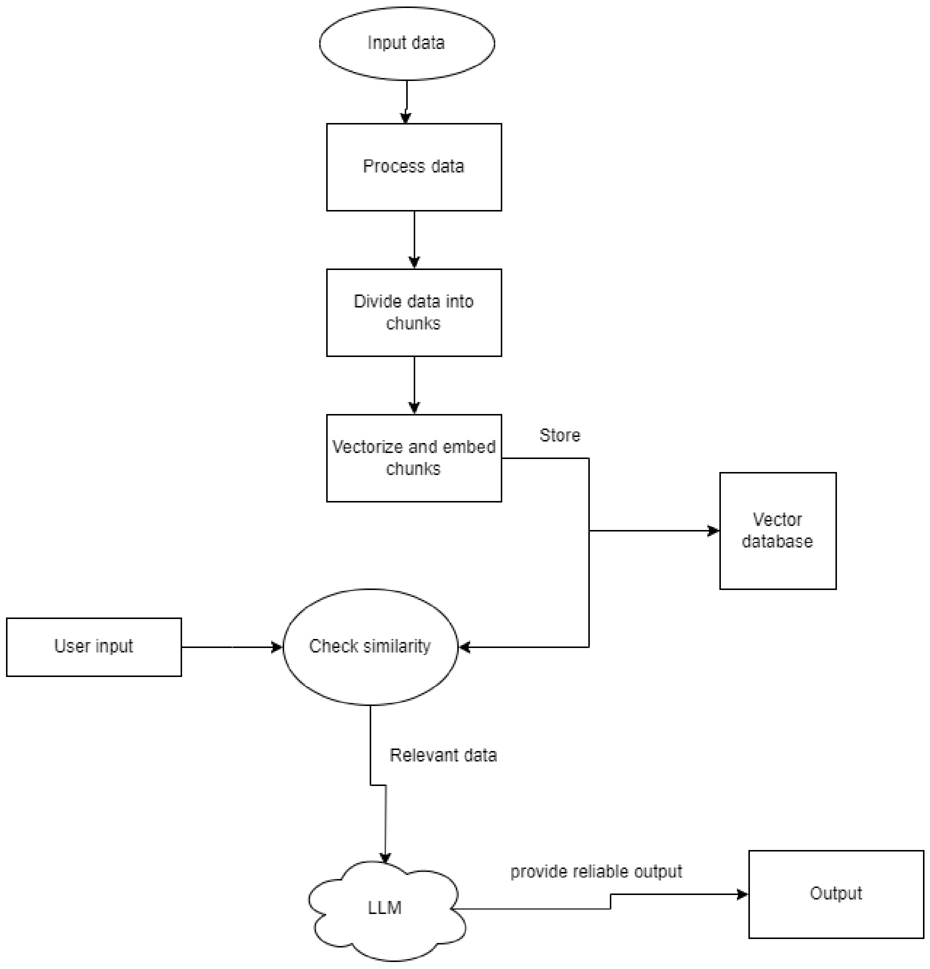
```
{
  "LoansDescription": {
    "BankName": "Smart Bank",
    "LoanTypes": [
      "Overdraft Facilities",
      "Housing Loans",
      "Education Loans",
      "Pensioner's Loans",
      "Personal Loans"
    ],
    "LoanSchemes": [
      {
        "LoanType": "Overdraft Facilities",
        "Description": "An overdraft is the ideal way to manage your cash flo
        "Features": [
          "An overdraft is quick and easy to arrange",
          "The cash is available when you need it",
          "You only pay interest on what you use, not on the full amount of y
          "Sole proprietors can apply for an overdraft or increase their limi
        ]
      },
      {
        "LoanType": "Housing Loans",
        "SubTypes": [
          {
            "SubLoanType": "Housing Loan Scheme",
            "Features": [
              "Purchase a land to construct a house later",
              "Purchase a land and to construct a house",
              "Construct a house in a land owned by the borrower/s",
              "Purchase a house/ partly constructed house/ condominium unit",
```

Feature Extraction

- The data which was divided into chunks was then vectorized.
- The vectors were embedded and stored in a vector database (A database which supports natural language queries).

Generating output

- The model is connected to GPT-3.5 turbo LLM using an API key.
- When an input is fed into the model the query string is vectorized and the model checks for similarity between the vectorized chunks in the database and the input query. Then the most suitable chunk is sent to the LLM and the LLM will generate a refined output to the user.



Strategies to improve efficiency.

- The previous input of the user is also appended to the current query to endure seamless to carry out a seamless conversation.
- Only the chat history up to a certain extend is considered to improve efficiency.
- By dividing the dataset into chunks improved performance significantly since the LLM will be provided with only the necessary and relevant data to produce the most reliable output to the end user.
- Vectorizing and embedding the chunks increased search performance of the model significantly.
- The use of a vector database instead of a simple relational database helped the model to integrate seamlessly with the data layer of the application.

Challenges Faced

Data Quality

Initial challenges included a text document without a proper format. Also, the file was of PDF format which made it even more difficult to extract data using conventional means.

Connecting to the LLM

The API key available was of a deprecated version which caused feature limitations and compatibility issues.

Feeding data to the vectorizer

The model had to be configured to properly capture data in JSON format which initially caused accuracy issues since the vectorizer did not support files in Json format.

Model accuracy

- The model provided reliable and readable outputs for the given queries and also gave fallback responses when an invalid argument is provided.