Chatbot Prototype Documentation

Overview

This document outlines the design, features, functionality, limitations, and potential improvements of a chatbot prototype created to respond to predefined financial queries based on analyzed data from 10-K filings of Microsoft, Tesla, and Apple. The prototype is implemented using Python and operates via a command-line interface.

Features and Functionality

Predefined Queries

The chatbot can respond to the following predefined financial queries:

- "What is the total revenue for [company] in [year]?"
- "How has the net income changed for [company] from [year] to [year]?"
- "What is the total asset value for [company] in [year]?"
- "What is the total liability value for [company] in [year]?"
- "What is the cash flow from operating activities for [company] in [year]?"

Data-driven Responses

Responses are generated based on the financial data extracted and analyzed from 10-K filings. The chatbot provides specific numerical values or percentage changes for the requested financial metric and company/year combination.

User-friendly Interface

The chatbot operates in a command-line interface, allowing users to input queries and receive corresponding responses. Clear instructions are provided for the expected query format to facilitate user interaction.

Error Handling

If the user inputs a query that is not predefined, the chatbot responds politely indicating its capability to only provide information on predefined queries.

Limitations

Predefined Query Set

The chatbot is limited to a specific set of predefined queries and cannot understand arbitrary financial questions or follow-up inquiries.

Static Data

Responses are based on static financial data extracted from the last three fiscal years' 10-K filings, lacking real-time data access or analysis capabilities.

Company and Year Constraints

The chatbot is restricted to providing information for Microsoft, Tesla, and Apple within the specified fiscal years available in the dataset.

No Natural Language Processing (NLP)

The chatbot does not employ NLP techniques, relying solely on exact matches of predefined query formats for responses.

Future Improvements

Natural Language Processing (NLP)

Integrating NLP techniques would enable the chatbot to understand and respond to more conversational and varied queries, enhancing user experience.

Dynamic Data Integration

Implementing mechanisms to fetch and analyze real-time financial data would keep the chatbot relevant and up-to-date.

Machine Learning (ML)

Utilizing ML algorithms could enhance response accuracy and enable the chatbot to learn and adapt based on interactions over time.

Multi-modal Interaction

Supporting multi-modal interaction (e.g., voice input/output) would improve accessibility and user-friendliness.

Expanded Knowledge Base

Enriching the chatbot's knowledge base with data from a broader range of companies, industries, and financial metrics would enhance its usefulness and versatility.

This documentation provides insights into the current capabilities and areas for future development of the chatbot prototype, serving as a foundation for further enhancements and iterations. Each section details key aspects of the chatbot's design and functionality, along with potential strategies for advancing its capabilities to deliver more intelligent and comprehensive financial analysis services.

In [ ]:

```python
import pandas as pd

# Load and preprocess the financial data from CSV
def load_data(file_path):
    data = pd.read_csv('D:\Internship Experience\BCG GenAI\FinancialData.csv')
    # Convert numeric columns from string with commas to float
    numeric_columns = ['Total Revenue', 'Net Income', 'Total Assets', 'Total Liabilities', 'Cash Flow from (
    for col in numeric_columns:
        data[col] = data[col].replace(',', '', regex=True).astype(float)
    return data

def get_response(user_query, data):
    try:
        query_parts = [part.strip() for part in user_query.split(',')]
        if len(query_parts) < 3:
            return "Invalid query format. Please provide the company name, year, and financial metric (e.g.,

        company = query_parts[0]
        year = int(query_parts[1])
        metric = query_parts[2].strip().lower()  # Extract the metric and convert to lowercase

        # Filter data based on company and year
        filtered_data = data[(data['Company'].str.strip().str.lower() == company.lower()) & (data['Year'] ==

        if len(filtered_data) == 0:
            return f"No data available for {company} in {year}."

        # Retrieve the requested metric value
        if metric == 'total revenue':
            value = filtered_data['Total Revenue'].values[0]
        elif metric == 'net income':
            value = filtered_data['Net Income'].values[0]
        elif metric == 'total assets':
            value = filtered_data['Total Assets'].values[0]
        elif metric == 'total liabilities':
            if 'Total Liabilities' in filtered_data.columns:
                value = filtered_data['Total Liabilities'].values[0]
            else:
                return f"No data available for {metric} of {company} in {year}."
        elif metric == 'cash flow':
            if 'Cash Flow from Operating Activities' in filtered_data.columns:
                value = filtered_data['Cash Flow from Operating Activities'].values[0]
            else:
                return f"No data available for {metric} of {company} in {year}."
        else:
            return f"Unsupported metric: {metric}"

        return f"The {metric} for {company} in {year} is {value}."

    except ValueError:
        return "Invalid year format. Please provide a valid year (e.g., 2022)."
    except Exception as e:
        return f"An error occurred: {str(e)}"
```

```python
def main():
    # Load financial data from CSV
    file_path = 'D:\Internship Experience\BCG GenAI\FinancialData.csv'
    data = load_data(file_path)

    # Chatbot interaction loop
    while True:
        user_query = input("Enter your query (or 'quit' to exit): ")
        if user_query.lower() == 'quit':
            break

        response = get_response(user_query, data)
        print(response)

if __name__ == "__main__":
    main()
```

```
Enter your query (or 'quit' to exit): Apple, 2021, Total Revenue
No data available for Apple in 2021.
Enter your query (or 'quit' to exit): Microsoft, 2023, Net Income
The net income for Microsoft in 2023 is 72361.0.
Enter your query (or 'quit' to exit): Apple, 2021,Total Revenue
No data available for Apple in 2021.
Enter your query (or 'quit' to exit): Apple,2021,Total Revenue
No data available for Apple in 2021.
Enter your query (or 'quit' to exit): Microsoft, 2023, Total Revenue
The total revenue for Microsoft in 2023 is 211915.0.
Enter your query (or 'quit' to exit): Apple, 2023, Total Revenue
The total revenue for Apple in 2023 is 121000.0.
```

In [ ]: