Guide to Build an Investment Strategy Chatbot in Streamlit

Ashutosh Agarwal | Ayush kumar Singh

SOC - Advanced Investment Advisor Chatbot

Introduction

This guide will walk you through creating a Streamlit-based chatbot that takes user input in the form of a paragraph, extracts relevant investment preferences, analyzes top stocks using the yfinance library, and provides tailored investment advice. We'll leverage Spacy for NLP and LLaMA for language model tasks.

1 Step 1: Set Up Your Environment

1.1 1.1 Install Required Libraries

To start, ensure you have Python installed. Then, install the necessary libraries using pip:

- Streamlit for creating the web interface.
- Spacy for natural language processing.
- yfinance for fetching stock data.
- transformers and torch for language model tasks with LLaMA.

Command:

```
pip install streamlit spacy yfinance transformers torch
```

1.2 1.2 Download Spacy Model

Spacy requires a language model to perform NLP tasks. Download the English model:

Command:

```
python -m spacy download en_core_web_sm
```

2 Step 2: Create the Streamlit Application

2.1 2.1 Set Up Streamlit Interface

- Create a new Python file, e.g., `app.py`.
- Import the necessary libraries: Streamlit, Spacy, yfinance, and transformers.
- Initialize the Spacy model and the LLaMA model.

2.2 2.2 User Input Section

- Add a text area in Streamlit for users to input their investment preferences.
- Use Streamlit's `text_area` function for this purpose.

3 Step 3: Process User Input

3.1 3.1 Extract Parameters

- Define a function to extract investment parameters from the input text.
- Use Spacy for Named Entity Recognition (NER) and pattern matching.
- Extract information such as investment goal, risk tolerance, investment amount, investment horizon, preferred sectors, and volatility tolerance.

3.2 3.2 Handle Default Values

- Ensure that if certain information is not provided by the user, default values are assigned.
- Common default values might include a medium-term investment goal, medium risk tolerance, a default investment amount, and horizon.

4 Step 4: Fetch and Analyze Stock Data

4.1 4.1 Fetch Top Stocks

- Define a function to fetch top 10 stocks from yfinance based on the user's preferred sectors.
- If no sector preference is provided, fetch the top overall 10 stocks.

4.2 4.2 Predict Stock Prices

- Define a placeholder function to predict future stock prices.
- Implement a time series model learned, for accurate predictions.
- Determine the prediction period based on the user's investment horizon.

4.3 4.3 Calculate Volatility

- Define a function to calculate the volatility of the stock prices.
- Use historical stock data to determine the percentage change and standard deviation.

4.4 4.4 Analyze Stocks

- Define a function to analyze the fetched stocks based on predicted prices and calculated volatility.
- Filter out stocks with higher volatility than the user's tolerance level.
- Compile a list of stocks expected to provide profit within the user's investment term.

5 Step 5: Generate Investment Advice

5.1 5.1 Summarize Analysis

- Generate a summary of the stock analysis including expected profit and volatility.
- Ensure the summary includes all relevant parameters extracted from the user's input.

5.2 5.2 Use LLaMA for Natural Language Output

- Use LLaMA to generate a more natural-sounding investment advice based on the summary.
- Initialize the tokenizer and model, then generate text using the model.

6 Step 6: Integrate and Run the Application

6.1 6.1 Combine All Components

- Ensure all the functions for extraction, analysis, and advice generation are integrated within the Streamlit application.
- Add a button in Streamlit to trigger the analysis and advice generation.

6.2 6.2 Run the Streamlit Application

- Save your `app.py` file.
- Run the Streamlit application from your terminal.

Command:

```
streamlit run app.py
```