Be earning in-depth property analysis and suggestion chatbot

Created by: Nguyen Quoc Thai

Date: 08/05/2024

Table of Contents

1. About

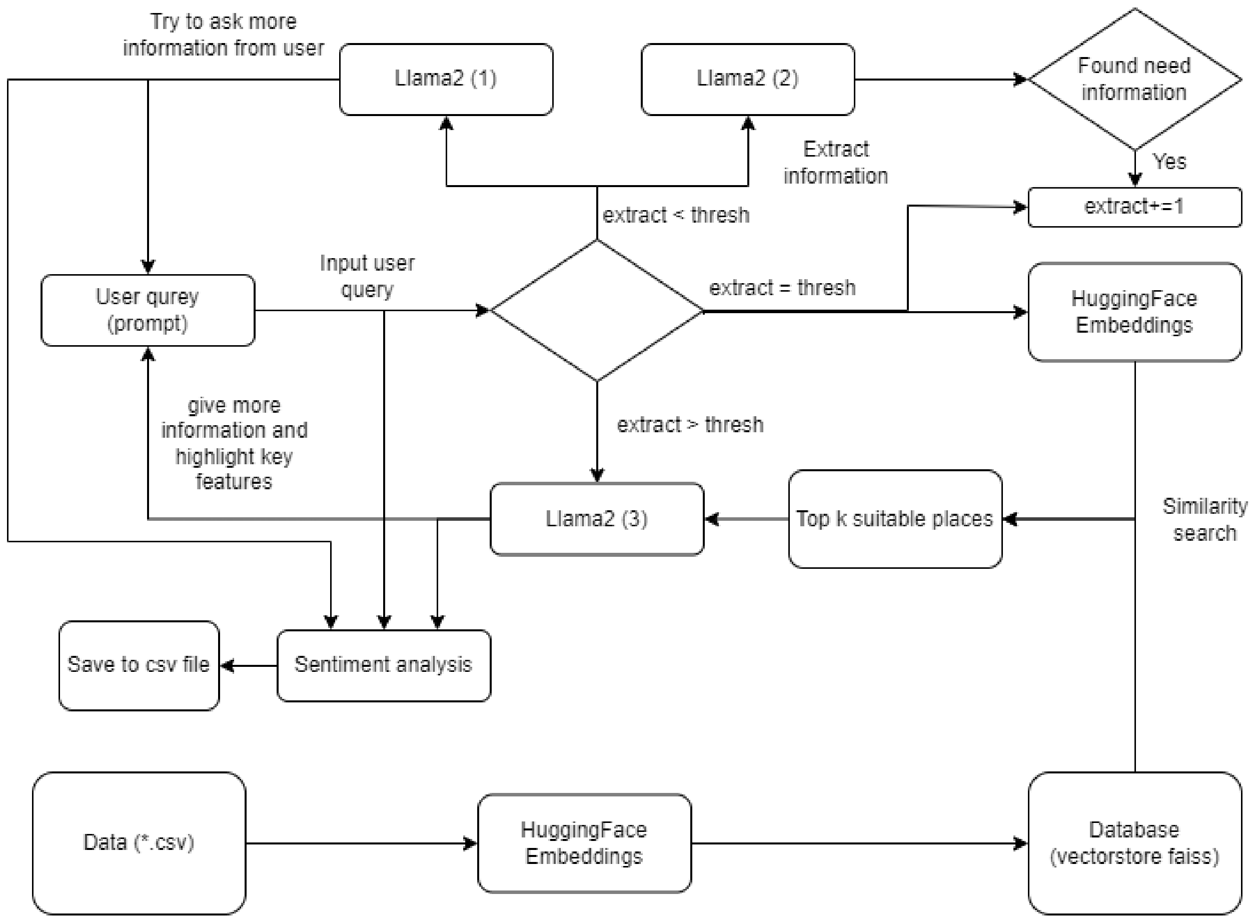
My property analysis and suggestion chatbot offers an in-depth exploration of various real estate options tailored to users' needs. Powered by the advanced language model Llama2, our chatbot engages users in meaningful conversations to identify their preferences regarding the city, budget, bedrooms, bathrooms, floors, facilities, noise levels, crime rates, and property specifications. Utilizing Streamlit for a user-friendly interface, our chatbot seamlessly guides users through the exploration process, providing detailed insights and recommendations for optimal property choices. Whether seeking a quiet neighborhood, top-notch schools, or proximity to essential amenities, our chatbot leverages state-of-the-art technology to deliver personalized and insightful property suggestions, ensuring a tailored real estate experience for every user.

2. Task description

Task	Status	Additional information
Extract key criteria from the user's statement	Done	Use Llama2 chat pretrained with system prompt
Refined Response: Generate a response that incorporates the LLM's analysis,	Done	Embed dataset with HuggingFaceEmbeddings then save with Faiss and use similarity search method in NLP
Allow the user to ask follow-up questions about specific properties identified by the LLM.	Done	Add user's preferences, suggestion places and some instruction to system prompt for further questions.
Integrate sentiment analysis to gauge user reaction and adapt future suggestions accordingly.	Done	Use sentiment analysis pretrained model to classify and save in csv type for analysis

3. Project Pipeline

By default, value of extract=0, thresh=7 (can be adjusted as need)



4. Algorithm explanation

Step 1:

Take the real estate data about the properties in CSV type. In this file, data is expressed with columns, for example in this code test I created a CSV file including 9 columns as Name, Location, Price, Bedroom, Bathroom, Noise levels, Crime rates, Property, and Additional Information.

Name: Name of the house/place.

Location: Name of the city where the property is located.

Price: price of this property.

Bedrooms, Bathrooms: Number of bedrooms and bathrooms in this place.

Noise levels, Crime rates: The level of noise and crime rates is described in 3 levels: Low, Medium, and High.

Property: Kind of this place: house, palace, cabin,…

Additional Information: Some description of this place.

Step 2:

From langchain, using the CSVLoader and HuggingFaceEmbeddings to read and embed the data for each row. Then save all the embedded data in vector type with Faiss. I used Faiss because:

Efficient Search: FAISS is an optimized library for searching vectors in high-dimensional spaces, particularly in nearest neighbor search tasks. When data is embedded into vectors and stored in FAISS, searching for nearest vectors becomes fast and efficient.

Memory Efficiency: FAISS can store embedded vectors efficiently, saving memory compared to traditional data storage methods.

Adaptable to Embedding Models: For embedding models like Word2Vec, GloVe, or LLM model, embedding vectors can be extracted and stored in FAISS easily.

Step 3:

With the dataset, I pre-defined 9 features that need to be asked/extracted from user call infor, and a threshold since we can not always get all need data from the user then threshold is the minimum feature we need to perform the analysis and suggestion task.

Step 4:

Take user query (prompt) from the User Interface of Streamlit, about the LLM model in this project I use the Llama2 chat with the input arguments as top_p=0.9, temprature=0.7, max_new_tokens=100, prompt and system_prompt. Except for prompt and system_prompt, the rest arguments can be adjusted on the user interface and used for all models in this project.

Step 5:

Use the Llama2 model, called this model as Model (1).

Model (1) input arguments include user_prompt (prompt from the user) and system_prompt (I defined this prompt to help the model understand and be able to do the task according to my intention). With system_prompt for Model (1), I want it to work as an in-depth property analysis and suggestion chatbot to help people find an ideal available house. Whenever chatting with the user, the chatbot tries to get as much information from the user about factors of the ideal place but does not ask them directly.

Step 6:

Whenever the user gives a prompt, use another Llama2 model, called this model as Model (2).

Model (2) input arguments include user_prompt (prompt from the user) and system_prompt. With system_prompt for Model (2), I want it to work as an extractor information tool from the input of the user and the pre-defined features. Model (2) checks if any information about the features is indicated by the user in their prompt. If yes, save them and plus 1 to the infor variable.

Step 7:

Check if infor var < threshold then turn back to Step 4 and retain the setting of Model before, else moving to the next step.

Step 8:

If infor var >= threshold, means we get enough information, then load the Faiss database and use the similarity_search with top k = 4 to return 4 places that has the features nearly the same as the user requirements.

Then use the Model (3), combine the user requirements and 4 suggestion places to generate a response to user with systemprompt as: a shortlist of relevant properties with brief descriptions and highlights of key features that match the user's preferences.

Step 9:

For further input from user use Model (3) to answer base on the user's preferences and 4 suggestion place to help find an ideal place.

Step 10:

For each pair of chatbot – user messages, use the sentiment analysis model from Hugging Face to perform the sentiment task for the user message and the classified label as the user's attitude of input message represent for this pair and be saved for future suggestions accordingly and analysis.

5. Potential improvements

Change another LLM (GPT open AI) or train a LLM from scratch for better replication.

Create a dataset to finetune the LLM in order to do the task Q&A and extract data information more effectively.

To extract the user’s need is quite hard since we can not control the way that the user inputs into the prompt. Therefore add the entity recognition model to extract the name of the place.

Use cloud and API to update the database in real-time for retrieval tasks.

Expand the field of features to match the user's preferences better.

Use the sentiment analysis results to modify the dataset and re-train LLM for better response generation.