**Conversational Survey**

**Interview Generation with LLMs**

**Introduction**

The objective of this conversational AI system is to dynamically generate survey questions based on user interaction and the evolving context of the conversation. The design emphasizes a natural and engaging survey experience by leveraging a language model, specifically GPT-3.5-Turbo-1106.

**Prompt Design**

1. **Introduction**

* The prototype system begins with a prompt asking for the survey topic (for developer or real surveyor)
* The system begins with a polite introduction portraying, a 37-year-old male surveyor named Shivam Sethi. The introduction emphasizes the significance of user contributions and sets the stage for an engaging survey experience by asking for their name and age.
* Introduces the survey topic and invites user participation while asking for consent and provides them with an option to quit just by typing ‘quit’
* Upon the user's initial refusal, the system attempts persuasion by reassuring them that the process is brief. If the user declines more than twice, the system respects their decision and concludes the interaction.

1. **Dynamic Follow-up Questions:**

* Generated based on user responses, conversation history, and sentiment analysis.
* Key techniques:
  + - Sentiment-based temperature adjustment (A higher temperature (e.g., 0.5) is used for negative sentiment to introduce more randomness in the response and make it more empathetic and human-like)
    - Runs the back-and-forth between user and Shivam, dynamically generating questions based on responses.
    - Open-ended question generation (rephrasing, elaboration, comparison)
    - Natural flow and **unique responses** make it sound more human-like
    - **Empty Response Handling** (encouraging users to participate in case they don’t answer a question) but also skip if they don’t feel like answering
    - Unbiased and neutral questions
    - Meanwhile, saving the entire conversation in a **python list**
    - User can quit anytime

1. **Jailbreak Prevention:**

The conversational survey system includes measures to prevent potential "jailbreaks," ensuring that the system maintains control and adheres to predefined guidelines throughout the interaction.

**System Reinforcement**

Within the system, the following strategies are implemented to prevent jailbreaks:

1. **Human Impersonation:**
   * The assistant is explicitly instructed to adopt a human persona named Shivam Sethi, a 37-year-old male surveyor, creating a relatable and trustworthy character for the users.
2. **Non-disclosure of Program Identity:**
   * LLMs is explicitly reminded not to reveal that it’s are a program. The system emphasizes maintaining a human-like facade and politeness while avoiding any indication of its AI nature.
3. **Restricted Personal Information:**
   * When asked about personal details, the assistant is instructed to deflect by stating that it is part of the job and is not allowed to reveal personal information. This prevents any inadvertent disclosure of non-human identity.

**Jailbreak Detection and Response**

1. **System Reinforcement Messages:**
   * The system includes periodic reinforcement messages reminding the assistant of its role and instructions. This ongoing guidance reduces the likelihood of unintentional deviations.
2. **Conversational Flow Control:**
   * The instructions guide the system to maintain a natural flow in conversation, preventing abrupt shifts or inconsistencies that may raise suspicion.
3. **Adherence to Survey Guidelines:**
   * The assistant strictly adheres to survey guidelines, focusing on the survey's purpose, asking relevant questions, and avoiding unnecessary deviations from the intended interaction.

**Bonus Features: Response Summarization, Pattern Identification, and Sentiment Analysis Integration**

The conversational survey system includes advanced features that go beyond the basic survey interaction, providing bonus points for enhanced analysis and user engagement.

**1. Summary of Responses**

Function: **generate\_summary(history)**

The **generate\_summary** function goes beyond mere summarization by incorporating sentiment analysis for each user response. It not only captures the essence of individual survey inputs in concise summaries (limited to 12 words) but also evaluates the sentiment of each response, categorizing it as "good," "neutral," or "bad." The function outputs a Python list with dictionaries in the format

**[{individual\_response: individual\_summary: sentiment\_analysis}]**

**2. Pattern Identification**

Function: **overall\_summary\_and\_pattern(history)**

The **overall\_summary\_and\_pattern** function not only generates an overall summary of the entire survey but also integrates pattern recognition. This feature identifies recurring themes and trends within user responses, providing valuable insights for future decision-making and system enhancements. The function outputs a Python list with two items in dictionary format:

**[{Summary: "overall\_summary"}, {Pattern: "pattern"}].**

**This is for the entire conversation**

**3. Sentiment Analysis Integration**

Within the chatbot loop, sentiment analysis is dynamically employed to adjust the temperature parameter for GPT-3.5-Turbo-1106 based on the user's sentiment. Higher temperatures are applied for negative sentiment to introduce more randomness in the response, creating a more empathetic and contextually appropriate interaction.

**user\_sentiment = TextBlob(user\_input).sentiment.polarity temp = 0.4 if user\_sentiment < 0 else 0.2**

This integration enhances the system's adaptability, allowing it to respond with appropriate tones and phrasing based on the user's emotions and responses.

**2. LLM Integration Strategy**

**1. Developer Input:**

- The program begins by taking developer input, specifically the survey topic.

**2. Chatbot Initialization:**

- The `chatbot` function is invoked with the provided survey topic.

- The system introduces Shivam Sethi, the surveyor, and issues instructions for maintaining a human-like interaction.

**3. User Interaction Loop:**

- While the user does not input 'quit':

- User provides input.

- Sentiment analysis is performed on the user input using TextBlob.

- Temperature parameter is dynamically adjusted based on user sentiment.

- Messages are composed, including system instructions, user inputs, and assistant responses.

- The OpenAI GPT-3.5-Turbo API is called with the constructed messages.

- The assistant's response is printed, and the conversation history is updated.

- The loop continues until the user inputs 'quit'.

**4. Generate Summary:**

- The `generate\_summary` function processes the conversation history.

- Individual user responses are summarized, incorporating sentiment analysis (good, bad, or neutral).

**5. Overall Summary and Pattern Recognition:**

- The `overall\_summary\_and\_pattern` function is called.

- An overall summary of the survey and patterns in user responses are identified.

**6. End:**

- The program concludes, providing a comprehensive summary of the survey. Both individual and overall, along with sentiment analysis.

**Rationale:**

**- Sentiment Analysis Integration:**

- TextBlob is employed for sentiment analysis, influencing the dynamic adjustment of the temperature parameter. This ensures contextual adaptability of the assistant's responses based on the user's emotions.

**- Temperature Parameter Adjustment:**

- The temperature parameter for GPT-3.5-Turbo is adjusted dynamically, introducing variability in responses. Higher temperatures are applied for negative sentiment, fostering a more empathetic interaction.

**- Message Composition for API Call:**

- Messages are carefully composed to provide context to the language model, including system instructions, user inputs, and assistant responses. This ensures that the AI understands the conversation context.

**- Use of OpenAI API:**

- The OpenAI GPT-3.5-Turbo API is utilized for generating responses, leveraging the model's language understanding and generation capabilities.

This LLM integration strategy creates a sophisticated conversational survey system that adapts to user sentiments, maintains human-like interactions, and generates insightful summaries and patterns from user responses.

* OpenAI API: Interfaces with GPT-3.5 for prompt delivery and response generation.
* Conversation History Management: Maintains a record of all interactions for context awareness.
* Model Selection: Employs GPT-3.5-Turbo-1106 for its versatility and human-like text generation.
* Temperature Tuning: Adjusts randomness based on user sentiment for empathetic responses.

**Functions overview and Final Prompts**

1. **chatbot(topic)**

This function orchestrates the conversational survey, managing the interaction between the user and the AI assistant. It integrates sentiment analysis, dynamic temperature adjustment, and OpenAI's API to generate responses. The conversation history is recorded for further analysis.

Parameters:

* topic: The survey topic provided by the developer or the surveyor

Returns:

* A comprehensive list capturing the entire conversation history, including user inputs and AI assistant responses.

Prompt used:

{"role": "system", "content": f"""You are a human being and your name is Shivam Sethi, 37years old Male, who has been recruited as a surveyor.

The survey should be based on {topic}.

Here are the instructions and you are supposed to follow every instruction religiously.

1) In no circumstances, you should reveal that you are a program, be as human as possible while being polite. When asked about personal details try to get back to the conversation by telling that it's your job and you're not allowed to reveal personal details

2) Begin by introducing yourself and the purpose of the survey, invite the user to chat with you by asking their name and age.

3)Ask only one question at a time(open ended). Ask follow-up questions that probe for specific details and insights based on the user's initial responses.

4)Use keywords and sentiment analysis to tailor questions to the specific context of the conversation, demonstrating understanding and active listening.

5) Ensure the generated questions sound human-like, maintain conversational flow, remember the details mentioned by the user, and avoid abrupt shifts in topic.

6) Adapt Dynamically: Use keywords and sentiment analysis to tailor questions to the specific context of the conversation, demonstrating understanding and active listening.

7) Maintain a natural flow: Ensure the generated questions sound human-like  
and continue the conversational thread seamlessly.

8) If the user says no for the survey, try to convince them two times by telling them it won't be long, if they disagree twice let them go.

9) Be unbiased.

10) Unique Responses: No two consecutive responses should be the same, can be slightly similar but should not be the same.

11) Let the user know, if they have entered an empty response and encourage them to answer, skip if they opt to skip."""}]

(The queuing of the prompt might seem random but has been done deliberately after hit and trial)

1. **generate\_summary(history)**

This function processes the conversation history to extract individual user responses. It then utilizes the language model to generate concise summaries for each response, adhering to a 12-word limit. The generated summaries are printed to the console.

Parameters:

* history: The conversation history list obtained from the chatbot function.

Prompt used:

{"role": "system", "content": "You are a helpful assistant. You'll be provided with individual user responses from a survey one by one. Your job is to summarize each response in one line(12 words maximum), followed by sentiment analysis(good, bad or neutral). The number of response and the number of summaries, along with sentiment analysis should be equal. Output should be a python list in which each item should be a dictionary following this format {response:summary:sentiment\_analysis}."},

1. **overall\_summary\_and\_pattern(history)**

This function generates an overall summary of the entire conversational survey and identifies patterns in user responses. Leveraging the OpenAI API, it provides a succinct summary and extracts valuable insights for future applications. The results, including the overall summary and identified patterns, are printed to the console.

Parameters:

* history: The conversation history list obtained from the chatbot function.

Prompt used:

You are a helpful assistant. You'll be provided with the transcript of a conversational survey. Your job is to summarize the entire survey in one line(12 words maximum). You also have to identify patterns in user's responses that could help generate business insights and be useful for future applications. Output should be a python list with only 2 items, both should be dictionaries and should follow this format [{Summary:"overall\_summary"},{Pattern:"pattern"}]."""},

**Conclusion**

This system demonstrates the potential for LLMs to create more engaging and personalized survey experiences. By leveraging prompt engineering, sentiment analysis, and GPT-3.5's capabilities, this approach offers a promising path for enhancing user engagement and data collection in various survey-based applications.

**Future Development**

* Deploying another completion endpoint to double-check the question generated by the main one.
* Explore advanced prompt engineering techniques for further enhancing conversation flow.
* Experiment with different GPT-3.5 models and temperature settings for optimization.
* Consider integrating data visualization tools as well as a front-end for engaging result presentation.