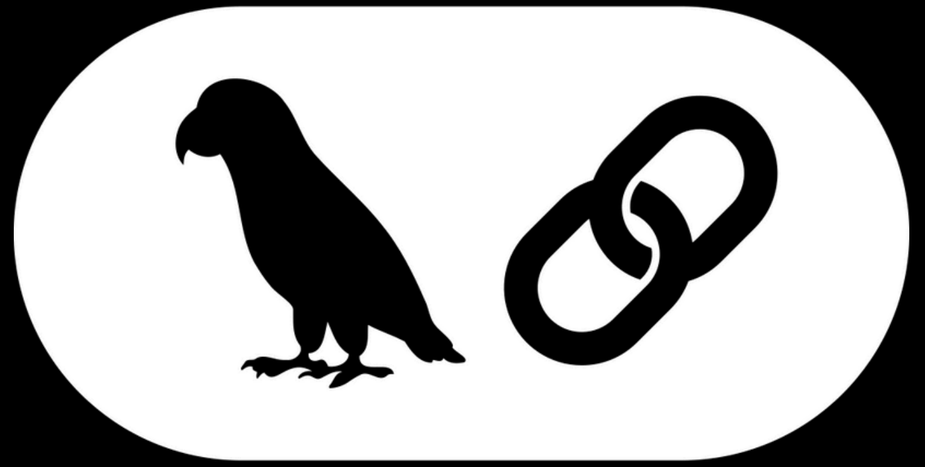
TEAM_OUTLIERS



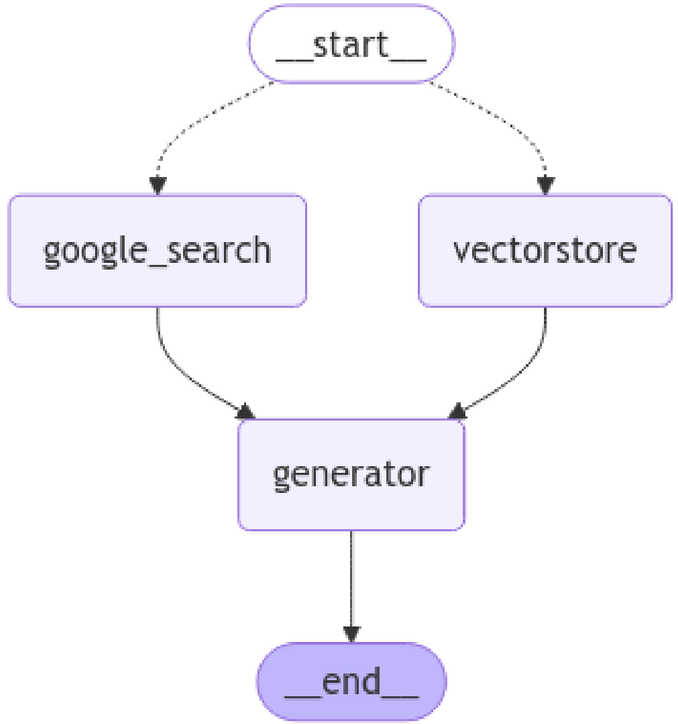
Langgraph based Insurance Agent

VIVEK KUMAR
9873432572
IIT BHU

GAURAV SINGH
70800 65387
IIT BHU

Workflow

- Audio Model - Whisper Turbo (for fast speech to text conversion)
- embedding model - all-MiniLM-L6-v2 (for vectorizing the data)
- database - astra vector database (for indexing all the doc chunks containing info related to insurance policy of max life insurance)
- Memory Saver - for deriving context from previous communication in the same thread
- Routing - query routing using llama - 3.1 8b instant (for routing query between vector database and google search
- search tool - tavily search tool (for searching info from google)
- final llm - llama 3.3 70b (for generating response using the retrieved info)
- text to speech - using gtts library
- gradio based web app



DETAILED WORKFLOW

1. Install Dependencies

- Libraries such as tavily-python, langchain_community, langgraph, whisper, gTTS, and gradio are installed.
- These libraries are necessary for API communication, text/audio processing, and the chatbot's workflow.

2. Configure Environment

- Environment variables like TAVILY_API_KEY and LANGCHAIN_API_KEY are set up to authenticate access to external services.
- The os.environ dictionary is used to store keys for APIs like Groq and LangSmith.

3. Speech-to-Text Processing

- Whisper is loaded as the transcription model (turbo mode with GPU support).
- Function: speech_to_text(audio_path)
  - Takes an audio file and transcribes its content into text.
  - Example: Converts /content/my name is will.m4a into a textual query.

4. Data Retrieval Setup

- A set of URLs related to Max Life Insurance is loaded using langchain.document_loaders.WebBaseLoader.
- Documents are split into smaller chunks using RecursiveCharacterTextSplitter for processing by the vector database.
- Example:
  - URL: "https://en.wikipedia.org/wiki/Max_Life_Insurance"
  - Processed into ~500-character chunks for efficient embedding and search.

5. Embedding Documents

HuggingFace Embeddings are used to vectorize the document chunks.

These embeddings are stored in a Cassandra database for fast similarity-based retrieval.

Cassandra Vector Store is set up to allow querying against these embeddings.

6. Query Routing

A routing mechanism (RouteQueryclass) determines where to send the user's query:

Vector Database: For specific queries related to insurance documents (e.g., "What is Smart Secure Plus Plan?").

Google Search: For generic web queries (e.g., "Who is Virat Kohli?").

LLM: For conversational or instructional queries (e.g., "Explain the benefits of term insurance.").

7. Query Execution

Based on the routing decision:

VectorStore: Retrieves relevant documents from the database.

GoogleSearch: Performs a search using Tavily APIs and retrieves content.

LLM: Processes the query using a fine-tuned LLaMA model to generate a conversational response.

8. Response Generation

The combined result from the data source (e.g., documents, search, or LLM output) is processed by a final response generator.

Template: A prompt template structures the output to ensure the response is clear and relevant.

9. Text-to-Speech

The response is converted into an audio file using Google's Text-to-Speech (gTTS).

Function: text_to_speech(text)

Saves the audio response as an .mp3 file.

Example: Converts "The premium of Smart Secure Plus Plan is X" into a playable audio file.

10. Real-Time Interaction with Gradio

Gradio Interface is used to provide a user-friendly GUI:

Input: Audio (speech) or text.

Output: Text response and audio playback.

Users interact in real-time, receiving both textual and verbal responses.

11. Stream Updates and StateGraph

LangGraph's StateGraphis used to define and control the workflow:

Nodes: Represent distinct actions (e.g., google_search, vectorstore, generator).

Edges: Define transitions between nodes based on the query type.

Example: A user's query is routed to vectorstore, processed, and passed to generator for response.

12. End-to-End Pipeline

The chatbot pipeline integrates all components:

Input: Audio or text query.

Speech-to-Text: Converts audio to text.

Routing: Determines the data source.

Query Execution: Retrieves or generates the response.

Text-to-Speech: Converts response to audio.

Output: Delivers text and audio to the user.

13. Examples of Queries

Specific Query: "What is the premium of Smart Secure Plus Plan?"

Routed to Vector Database.

General Query: "Who is Virat Kohli?"

Routed to Google Search.

Instructional Query: "Explain benefits of term insurance."

Routed to LLM.

14. Dynamic Query Handling

Users can interactively input queries, and the system dynamically processes responses using graph.stream for real-time updates.

What Users Are Saying About Our Chatbot

Rahul Sharma, Insurance Agent

"This chatbot has revolutionized how I interact with clients. It's fast, accurate, and handles both voice and text queries seamlessly!"

Ananya Mehta, Customer

"I asked about retirement plans and received clear, concise answers in seconds. The voice responses made it feel so natural."

Arjun Gupta, Small Business Owner

"The chatbot helped me compare insurance plans effortlessly. It even explained complex terms in a way I could understand!"

Priya Nair, IT Consultant

"The real-time interaction with audio responses is a game-changer. It's like having a personal assistant available 24/7."

Transform Your Customer Experience with AI

(Empower your business with our real-time voice-to-voice chatbot solution.)

Key Takeaways:

Seamless Integration: Combines speech-to-text, text-to-speech, and smart query routing.

Efficient Query Handling: Delivers instant and accurate responses.

Enhanced Engagement: Provides human-like conversations with text and audio.

"Your customers deserve the best. Let our chatbot make it happen!"