**Chatbot for RHCP-InterfaithRise: Development and Evaluation Methodology**

**Introduction**

This document outlines the methodology for developing a chatbot for RHCP-InterfaithRise, based on the paper titled "Chatbot Implementation" from arXiv (ID: 2405.08120). The chatbot integrates advanced AI techniques, including speech recognition capabilities, to enhance user interaction and support.

**Chatbot Development Methods**

**Overview**

Chatbots can be designed using various methods, including flow-based and AI-based approaches. The choice of technology significantly impacts the chatbot's performance and quality, necessitating careful selection during the design and development phases.

**Flow-Based or No-Code Chatbots**

1. **No-Code Chatbot Builders:**
   * **What Are They?** A no-code chatbot builder is a software tool that enables the creation of chatbots without requiring coding skills. These platforms provide a user-friendly interface, allowing individuals or teams without technical expertise to build and deploy chatbots.
   * **How Do They Work?** No-code tools offer visual interfaces where users can interact with pre-built templates or modules. Users can drag and drop components to create chatbots tailored to specific business needs.
   * **Benefits:**
     + **Accessibility:** Empowers non-technical professionals to create applications without deep programming knowledge.
     + **Agility:** Enhances the ability to build and implement chatbots swiftly based on specific requirements.
     + **Ease of Use:** Simplifies the chatbot creation process, making it accessible to a broader audience.
   * **Shortcomings:**
     + **Limited Customization:** May have limitations in terms of customization compared to custom-coded solutions.
     + **Complex Scenarios:** Handling complex chatbot scenarios might be challenging without coding skills.
     + **Dependency on Templates:** Often rely on pre-built templates, which may not cover all use cases.
   * **Popular No-Code Chatbot Builders:**
     + **Tidio:** Visual chatbot builder with a drag-and-drop editor, suitable for small and medium businesses.
     + **HubSpot:** Seamlessly integrates chatbots with other HubSpot services.
     + **Chatfuel:** Ideal for social media chatbots, with a free trial available.
     + **Pandorabots:** Known for AI chatbots.
     + **ManyChat:** Supports omnichannel chatbots.
     + **ChatBot:** Provides chatbot analytics.
     + **WATI:** Specializes in WhatsApp chatbots.
     + **Outgrow:** Useful for collecting customer information.
     + **TARS:** Offers comprehensive onboarding tutorials.
     + **Aivo:** Known for its ease of use.

**Code-Based Chatbots**

* **Customization:** Custom-coded chatbots allow for extensive customization using advanced techniques like Retrieval-Augmented Generation (RAG) and fine-tuning models for specific tasks.

**Assessment of Chatbot Effectiveness**

**Evaluation Methods**

1. **Surveys:**
   * Collect feedback from students and educators regarding their experiences with the chatbot.
2. **Experiments:**
   * Test chatbots in controlled settings to measure their impact on user engagement and satisfaction.
3. **Evaluation Studies:**
   * Assess acceptance, motivation, and usability through detailed studies.

**Interaction Styles**

Research distinguishes between user-driven and chatbot-driven interactions:

* **User-Driven:** Prioritizes user input, offering more flexibility.
* **Chatbot-Driven:** Involves automated and guided conversations for a structured interaction.

**Ethical Considerations**

* **Privacy and Personal Information:** Ensure user data is protected and privacy is maintained throughout the interaction.

**Two Main Phases of Chatbot Development**

**Context Retrieval**

1. **Retrieval Process:**
   * Obtain pertinent information from external data sources to establish context for responses.
   * **Data Curation:**
     + Use web crawlers to gather relevant data.
     + Transform external data sources using embedding models and vector databases.
   * **Embeddings:**
     + Functions that map raw data to low-dimensional vector representations while retaining important semantic information.
   * **Vector Database:**
     + Stores data as high-dimensional vectors, supporting complex and unstructured data for fast and accurate retrieval.
   * **Implementation:**
     + Utilize the text-embedding-3-large model managed through API calls.
     + Store vectors in Chroma DB, an in-memory vector database.
     + Employ LangChain's vector store-backed retriever technique using methods like Maximum Marginal Relevancy (MMR) and Similarity Search.

**Completion or Response Generation**

1. **Response Generation:**
   * Utilize a GPT-based LLM (e.g., OpenAI’s GPT-3.5-turbo) for generating responses.
   * Input consists of retrieved document chunks and the user prompt.
   * Generate accurate and relevant responses using the generator model through API calls.

A diagram of a process flow

Description automatically generated

**Evaluation of System Performance**

**Quantitative Evaluation**

* **RAGAS Framework:**
  + Assess the RAG pipeline using metrics like context precision, context recall, faithfulness, and answer relevance.
  + **Context Precision:** Evaluates the Signal-to-Noise Ratio (SNR) of retrieved context.
  + **Context Recall:** Assesses the ability to retrieve all relevant evidence.
  + **Faithfulness:** Measures factual accuracy of generated answers.
  + **Answer Relevance:** Evaluates the relevance of generated answers to the questions.
  + **RAGAS Score:** A singular measure representing the harmonic mean of the four metrics, ranging from 0 to 1.

**Usability Assessment**

* **System Usability Scale (SUS):**
  + Conduct a satisfaction survey using SUS with a panel of 50 students.
  + Collect feedback on usability, resulting in an average SUS score of 67.75, indicating satisfactory usability.

**Implementation Details**

* **Data Curation:**
  + Employ a multi-thread web crawler with the Scrapy Python library.
  + Manually select important HTML div tags to remove noise.
  + Export data to JSON files and consolidate them into a master JSON file for comprehensive retrieval.
* **Preprocessing:**
  + Use Recursive Character Text Splitter strategy with a chunk size of 8000 and overlap of 1200 characters.
  + Apply embedding functions using OpenAI’s text-embedding-3-large model and store vectors in Chroma DB.
* **Context Retrieval:**
  + Utilize LangChain’s vector store for similarity search.
* **Response Generation:**
  + Leverage OpenAI’s GPT-3.5-turbo model.
* **Development Framework:**
  + Built with Django using Python.
  + Front-end developed with HTML, CSS, and JavaScript.
  + Features include user sign-up, login, query management, and conversation history.
  + Deployed via a third-party cloud service for accessibility.

**Limitations**

* **Speech Recognition:**
  + Limited capabilities and integration.
* **Multilingual Support:**
  + Currently limited to a few languages.
* **Hallucinations:**
  + Occasional generation of incorrect or nonsensical responses.
* **Limited Output Tokens:**
  + Constraints on the number of tokens the model can generate.
* **Context Window:**
  + Restricted context window size affecting response relevance.

Map reduce, Document chain approach from langChain

Literature revised

[2405.08120 (arxiv.org)](https://arxiv.org/pdf/2405.08120) CHATBOT FOR WISCONSIN UNIVERSITY

[Build a simple RAG Chatbot with LangChain | by Kong Nopwattanapong | Credera Engineering (medium.com)](https://medium.com/credera-engineering/build-a-simple-rag-chatbot-with-langchain-b96b233e1b2a)