AI Chat Application

Prototype Project Report

MeakuAI

Aayush Shrestha

Gabe Compton

Matei Vitalaru

Table of Contents

II. Introduction

The field of artificial intelligence has made significant progress in recent years, particularly in natural language processing and generative AI. As these technologies continue to evolve, there is an increasing need for robust evaluation frameworks to assess and optimize AI-powered chat applications. This project aims to address this need by developing a comprehensive evaluation framework for a knowledge-based AI chat application. The motivation behind this project stems from the growing importance of AI chatbots in various industries, including effective customer service. As these systems become more prevalent, it is crucial to ensure their responses are accurate, relevant, and efficient. By creating a standardized evaluation framework, we can systematically assess and improve the performance of AI chat applications, ultimately enhancing user experience and the practical utility of these systems.

III. Background and Related Work

The development of AI-powered chat applications has its roots in early natural language processing (NLP) systems and expert systems. Over the years, advancements in machine learning, particularly deep learning, and transformer models, have significantly improved the capabilities of these applications.

Several key areas of research and development are relevant to this project:

1. **Embedding Techniques**: Word embeddings, such as Word2Vec, GloVe, and more recently, contextual embeddings like BERT, have revolutionized how machines understand and represent text. These techniques allow for more nuanced and context- aware processing of natural language.
2. **Language Models**: The advent of large language models like GPT has solidified a foundation in the field of AI. These models have demonstrated impressive capabilities in generating human-like text and answering complex queries.
3. **Information Retrieval**: Modern vector search methods, form the backbone of knowledge-based question answering systems. Research in this area focuses on efficiently storing and retrieving relevant information from large datasets.
4. **Evaluation Metrics**: Various metrics have been developed to evaluate the performance of language models and chatbots. These include BLEU and ROUGE scores for measuring text similarity, as well as more specialized metrics for assessing chatbot performance.
5. **Vector Databases**: Such as pgvector, has enabled efficient storage and retrieval of high-dimensional embedding vectors, which is crucial for scaling knowledge-based AI systems. To successfully complete this project, our team will need to develop new technical knowledge and skills in several areas:
    a. Advanced NLP techniques and embedding methods.
    b. Data preprocessing and augmentation techniques for unstructured text data
    c. Implementation and fine-tuning of language models.
    d. Vector database management and optimization.
    e. Design and implementation of evaluation frameworks for AI systems

IV. Project Overview

The primary objective of this project is to build a comprehensive evaluation framework for an AI-powered chat application that answers questions based on a knowledge base of unstructured data. The project will involve several key components:

1. Development of a Simple Chat Application:
   - Integrate the application with a knowledge base composed of unstructured data from various sources, including webpages, PDFs, and other text documents.
   - Integrate AI to process user queries and map it to vector database.
2. Knowledge Base Handling:
   - Utilize web scrapers like Beautiful Soup to ingest and clean unstructured data.
   - Implement embedding creation methods to convert text data into vector representations.
   - Utilize pgvector to efficiently store and query the embeddings.
3. Parameter Identification and Optimization:
   - Investigate and document various parameters that influence response quality, including embedding creation parameters, model variables, knowledge base variables, and system performance parameters.
   - Develop methods to systematically adjust and test these parameters.
4. Evaluation Framework:
   - Incorporate automated metrics (e.g. BLEU, ROUGE) and human-in-the-loop testing for a holistic evaluation.
   - Evaluate embedding quality, model performance, response relevance, and system efficiency.

V. Client and Stakeholder Identification and Preferences

1. **Primary Client**: MeakuAI
2. **Needs**: A robust evaluation framework for their AI-powered chat application.
3. **Preferences**: Emphasis on balance between response quality, and system latency.
4. **Project Mentor and Co-mentor needs**: Clear project progress update every other week thorough documentation.

VI. Project Requirements

Pre-condition

- The user accesses the website where the AI assistant is integrated.
- The website must allow for the AI model to be integrated.

Post-condition

- The user can ask questions specific to the website's content.
- The AI provides relevant and accurate answers based on the website data.

Basic Path

- The user opens the website.
- The user interacts with the AI by entering a question or query.
- The AI processes the question and retrieves information relative to the query from the website.
- The AI presents the answer to the user in a user-friendly format, referencing where it retrieved the data.

Alternative Path

- The user asks a question that the AI cannot answer (e.g., unsupported question types or no relevant data).
- The AI responds with an error message or an alternative suggestion, such as asking for more clarification or directing the user to a relevant section of the website.

VII. System Requirements Specifications

1. **Use Cases**
    I. **User Input**
    - **Description:** The AI must allow user entered queries in which the program will then proceed to answer via keyword detection
    - Scenario: User is on MeakuAI page
    - Given: I am on the MeakuAI page
    - When I enter a query in the chat box
    - And I click “send”
    - Then MeakuAI will register keywords from query
    - And I should see results most applicable to my search query

    II. **Viewing AI Results**
    - **Description:** The AI should be able to display accurate results to the chat box along with the source where the information was found
    - Scenario: User has submitted search query
    - Given: I have clicked “send” after typing my question
    - When I view the AI response
    - Then I should see an answer/response from MeakuAI
    - And I should see the source from where the information was pulled

    III. **Company Information Inquiry**
    - **Description:** The AI should be able to answer specific questions about the company.
    - Scenario: User is on the MeakuAI page
    - Given: I am on the MeakuAI page
    - When I ask specifics about MeakuAI
    - Then I should see the specifications I asked for
    - And I should see relevant sources if any

    IV. **Customer Support**
    - **Description:** The AI should be able to answer difficulties users may run into when using MeakuAI
    - Scenario: User is on MeakuAI page
    - Given: I am on the MeakuAI page
    - When I ask for help using MeakuAI or product description for ease of use
    - Then I should see relevant actions one can take based off the query
    - And see relevant product capabilities for my issue

2. **Functional Requirements**
    I. **Question Comprehension and Contextual Retrieval:**
    - **Description:** The AI must understand natural language queries related to the content of any specific website and retrieve information based on the context of the website.
    - **Source:** User experience goals.
    - **Priority:** Essential functionality.

    II. **Website Data Parsing:**
    - **Description:** The AI must be capable of extracting, indexing, and understanding content from any website.
    - **Source:** Internal technical research.
    - **Priority:** Essential functionality.

    III. **Cross-Site Usage:**
    - **Description:** The AI system should be able to work on any website

with minimal modifications and adapt to the structure of various websites.
- **Source:** Client specifications for cross-website functionality.
- **Priority:** Essential functionality.

IV. **Real-time Answering:**
- **Description:** The AI must provide answers in real-time or with minimal latency to ensure a smooth user experience.
- **Source:** Usability testing and performance benchmarks.
- **Priority:** Essential functionality.

3. **Non-Functional Requirements**

I. **Scalability:**
- **Description:** The system must scale efficiently to handle multiple users and high-traffic websites without significant degradation in performance.
- **Priority:** High.

II. **Data Privacy and Security:**
- **Description:** The AI must handle user queries and website data securely, following best practices for data encryption and protection to ensure no sensitive information is leaked.
- **Priority:** High.

Related Requirements
- **Website Content Indexing**
- **Natural Language Processing (NLP)**
- **Real-time Query Processing**
- **AI Model Training**

VIII. System Evolution

The development of our AI-powered chat application is based on several key assumptions regarding the tools and techniques that will be used. These assumptions may evolve due to changes in technology, user needs, or unforeseen limitations:

1. **Retrieval-Augmented Generation (RAG) Framework**: We assume that a RAG application is the optimal approach for providing accurate and contextually relevant responses. This involves using a combination of a pre-trained language model and a vector store for efficient information retrieval. We anticipate potential challenges if the RAG framework proves inefficient due to the unstructured nature of the data provided or if the retrieval accuracy is insufficient.
2. **Vector Store Utilization**: The system assumes the use of a vector store, such as pgvector, for indexing and searching through embeddings generated from the knowledge base. This depends on the efficiency of the vector store in handling a large volume of unstructured/semi-structured data and providing low-latency responses. Should performance issues arise, such as slow retrieval times or limitations in scalability, we may need to switch to a different vector storage solution or optimize our current choice by partitioning data more effectively or upgrading the underlying hardware, which could impact the system's architecture and deployment strategy.
3. **Web Scraping for Knowledge Base Construction**: We rely on web scraping tools like Beautiful Soup to collect unstructured data from the knowledge base to create a semi-structured database. This assumes that the data sources are stable and

accessible, and that our scraping scripts can reliably extract the required information. If any of the data sources undergo significant changes in structure, we may need to redesign the web scraping approach. This would require additional development time and could introduce new risks related to data quality and access

4. **Model Adaptation**: The language model we use for generating responses assumes a certain level of pre-existing training that can be easily adapted to our use case with fine-tuning. We anticipate changes if the model cannot generalize well enough to the specific questions derived from the unstructured data, necessitating more extensive training or perhaps even a switch to a different model architecture (e.g., moving from GPT-based models to more specialized domain-specific models). This would require additional computational resources and an extended training phase.
5. **Latency Considerations**: A key assumption is that the combination of retrieval (from the vector store) and response generation (from the language model) will meet latency requirements specified by the client. If the combined retrieval and generation process introduces unacceptable delays, we may need to implement optimizations, such as using a caching layer for frequently asked questions or moving to a more powerful deployment environment, which could have implications for both cost and architecture.
6. **Scalability and User Load**: The system is designed based on the assumption that the anticipated user load will be manageable with the current deployment setup. If user demand exceeds expectations, scaling up may be required. This could involve adding more nodes for the vector store, using a distributed deployment for the model, or transitioning to a different cloud architecture to handle increased requests. Any of these changes could affect both the system's cost efficiency and its maintenance complexity.

6. **Risk Points**:
    - **Vector Store Scalability**: Issues related to the scalability or performance of the vector store may necessitate a move to alternative storage solutions or sharding strategies.
    - **RAG Inefficiency**: If the RAG approach struggles with the provided data type, we may need to reconsider our retrieval strategy, leading to changes in core algorithms.
    - **Web Scraping Challenges**: Changes in data source structures, access restrictions, or anti-scraping measures may require re-engineering the data collection process, potentially introducing delays and impacting data quality.
    - **Model Adaptation Failures**: Inadequate model performance on the domain-specific data could require more intensive fine-tuning or the selection of an entirely new model.
    - **Latency Concerns**: Meeting low-latency requirements may require a caching mechanism or deployment upgrades, affecting the cost and complexity of the solution.

IX. Glossary

- **AI** (Artificial Intelligence): The simulation of human intelligence processes by machines, especially computer systems.
- **NLP** (Natural Language Processing): A subfield of AI that deals with the interaction between computers and humans using natural language.
- **Embedding**: A representation of a word, sentence, or document as a vector of real numbers in a continuous vector space.
- **Vector Database**: A database that stores and queries high-dimensional vectors, often used for similarity search in machine learning applications.

- **pgvector**: An open-source vector similarity search for PostgreSQL.
- **Transformer**: A deep learning model architecture used primarily in NLP tasks.
- **BERT** (Bidirectional Encoder Representations from Transformers): A transformer-based machine learning technique for NLP pre-training developed by Google.
- **GPT** (Generative Pre-trained Transformer): A type of large language model that uses deep learning to produce human-like text.
- **BLEU** (Bilingual Evaluation Understudy) Score: An algorithm for evaluating the quality of text which has been machine-translated from one natural language to another.
- **ROUGE** (Recall-Oriented Understudy for Gisting Evaluation) Score: A set of metrics used for evaluating automatic summarization and machine translation.