Project Report: GenAI Document Generation Bot

1. Introduction

The GenAI Document Generation Bot is an AI-powered tool designed to automate document creation for customer service applications. The bot generates various types of documents—such as reports, summaries, and letters—based on user inputs, leveraging natural language generation (NLG) techniques. This project aims to develop a functional prototype for generating customized documents by using pre-defined templates, external data sources (such as customer service FAQs and product manuals), and a fine-tuned language model. The bot is designed to provide users with an intuitive interface to specify the document type, provide input details, and automatically generate corresponding documents in a user-friendly format.

2. Architecture Design

2.1. Model Selection

For the NLG component of the bot, we selected Google Gemini (an advanced AI model, similar to GPT-3), which is well-suited for generating coherent and contextually accurate text. Gemini has been integrated via the `google.generativeai` library, providing powerful capabilities for content generation.

To handle natural language understanding (NLU), we leverage language embeddings (via the `HuggingFaceEmbeddings` class) to analyze and extract relevant context from large-scale data sources, such as customer service FAQs and product manuals. These embeddings help the bot understand user queries and align them with relevant content from the knowledge base.

2.2. Data Sources

The knowledge base of the bot is built from multiple sources:

- **Predefined PDFs**: Relevant documents (e.g., product manuals, customer service guides) are loaded and processed using the `PyPDFLoader` class.
- **Web Scraping**: We scrape customer service-related web pages to gather additional relevant information, using the `BeautifulSoup` library to extract content from HTML.
- **User Input**: The bot incorporates dynamic user inputs, which are processed in conjunction with the existing knowledge base to generate accurate responses.

2.3. System Components

The architecture consists of the following components:

1. **Document Loader**: Loads PDFs and web data into the bot's knowledge base.
2. **Text Splitter**: Breaks down documents into manageable chunks for processing.

3. **Embeddings and Vectorstore**: Embeds the textual content into vector space for fast retrieval.
4. **Template Engine**: Handles dynamic generation of documents using pre-defined templates and user inputs.
5. **GenAI Model Integration**: Leverages the Google Gemini model for content generation based on the query and context.

3. Prototype Implementation

3.1. Input Handling

To handle user input, the bot offers a simple and intuitive interface through the Streamlit library. Users can:

- Select the document type (e.g., report, letter, summary).
- Provide query-specific inputs (e.g., recipient name, document title, subject).
- Specify any additional metadata fields, such as date and context.

These inputs are captured via Streamlit widgets, and the data is used to fill in pre-defined templates.

3.2. Document Templates

For each document type, a corresponding template is created. These templates are stored as plain text or markdown files and are filled dynamically based on the user input. For example:

- **Report Template**: Contains placeholders for the report title, author, and sections.
- **Letter Template**: Includes placeholders for recipient name, subject, and body content.
- **Summary Template**: Focuses on summarizing key points based on a given query.

The templates are rendered using the Jinja2 template engine, allowing for flexible text generation.

3.3. Generation Logic

The bot generates documents by filling in templates with content extracted from the knowledge base. The process involves:

1. **Retrieving Relevant Context**: The bot uses the vectorstore and embeddings to fetch documents or segments that are relevant to the user's query.
2. **Template Population**: The retrieved context is merged with the user's inputs and inserted into the corresponding document template.
3. **Content Generation**: Using the fine-tuned Gemini model, the filled template is transformed into a coherent document.

3.4. Document Generation & PDF Creation

Once the document is generated in markdown format, the bot can convert the result into a downloadable PDF file using the `reportlab` library. The `generate_pdf` function takes the

markdown content and formats it into a structured PDF document, which can be downloaded by the user.

3.5. User Interface

The user interface is built using Streamlit, a popular framework for creating interactive web applications. The interface allows users to:

- Select the document type.
- Provide relevant metadata fields.
- Enter a query or additional information.
- View the generated document.
- Download the final document as a PDF.
- Provide feedback on the document's quality (coherence, relevance, grammatical correctness, etc.).

3.6. User Feedback and Improvement

User feedback is an essential part of the system. After generating a document, the bot collects feedback on various quality metrics (e.g., coherence, grammar, relevance, satisfaction). This feedback is used to iteratively improve the document generation process. For example, it can be used to fine-tune the underlying language model or improve the document templates.

4. Evaluation of the Bot's Performance

4.1. Quality Metrics

To evaluate the bot’s performance, several metrics are defined:

- **Coherence**: Measures how logically consistent the generated document is.
- **Relevance**: Assesses how well the document addresses the user's query.
- **Grammatical Correctness**: Evaluates the text for proper grammar and syntax.
- **Satisfaction**: An overall user rating based on the quality of the document.

These metrics are captured through a feedback mechanism that allows users to rate the document.

4.2. User Feedback

User feedback is collected using sliders on Streamlit, where users can rate the generated document on a scale from 1 (Poor) to 5 (Excellent). The collected feedback is printed for analysis and can be used for continuous improvement of the system.

4.3. Challenges

- **Model Fine-Tuning**: Although pre-trained models like Gemini and GPT-3 provide powerful NLG capabilities, fine-tuning them for specific document types (e.g., reports, letters) can be challenging due to the need for domain-specific data.

- **Data Quality**: Ensuring the knowledge base is up-to-date and relevant is critical. Web scraping can be inconsistent, and extracting useful content from diverse sources (e.g., PDFs, websites) requires effective parsing techniques.
- **User Interface**: While the system is user-friendly, some users may find it difficult to understand the relationship between document types, templates, and metadata fields.

4.4. Future Improvements

- **Template Customization**: Allowing users to create and customize their templates for more flexibility.
- **Model Enhancement**: Fine-tuning the language model with more domain-specific data to improve the relevance of the generated content.
- **Automatic Feedback Incorporation**: Using user feedback to automatically adjust the model's generation capabilities, potentially through active learning techniques.

5. Conclusion

The GenAI Document Generation Bot provides an effective solution for automating document creation in customer service applications. By combining pre-trained AI models, external knowledge sources, and flexible document templates, the bot can generate high-quality documents based on user inputs. The system's architecture ensures scalability and ease of use, while the feedback mechanism allows for continuous improvement. With further enhancements and fine-tuning, the bot can be adapted to a wide range of document generation tasks.

6. Submission

- **Code**: The code is well-documented and provided in a Jupyter notebook format for ease of review.
- **Report**: This detailed report explains the approach, implementation, and evaluation metrics.

Assumptions:

- The bot assumes that user inputs are in a structured format, allowing it to extract relevant details.
- The knowledge base consists of a fixed set of data sources (e.g., predefined PDFs and web scraping results).

Challenges Faced:

- Handling large and diverse document templates while ensuring that they are filled correctly with user input.
- Ensuring high relevance and accuracy of the generated documents by carefully tuning the model.
- Transitioning from Ollama to Gemini+HuggingFace