An Internship Report on

Offline HR Pdf Chatbot

Submitted in fulfilment for
the **project work/practical training**

Submitted by **T T K Urshitha Sai**

Under the guidance of **Shri. J.V. Subramanyam (DCE)**

Nuclear Power Corporation of India Limited
(NPCIL - HQ)
Vikram Sarabhai Bhavan
Mumbai - 400094

ACKNOWLEDGEMENT

I am extremely grateful to everyone who supported and guided me throughout my internship at NPCIL. First and foremost, I would like to express my heartfelt gratitude to my Internship Supervisor, Shri. J. V. Subramanyam, DCE, IT group, for his unwavering guidance, constant support, and expert advice during this project. His insights and encouragement were instrumental in overcoming challenges and enhancing the quality of my work.

I would also like to sincerely thank my fellow interns for their camaraderie and collaboration, which made this journey both enjoyable and enlightening. Additionally, my gratitude goes to all the members of the IT group, whose coordination, cooperation, and provision of essential infrastructure ensured a smooth and successful internship experience.

Thank you all for your continuous support, encouragement, and guidance.

TABLE OF CONTENTS

I. Organisation Information

राष्ट्र की सेवा में परमाणु
एनपीसीआईएल
NPCIL
ATOMS IN THE SERVICE OF THE NATION
न्यूक्लियर पावर कॉर्पोरेशन ऑफ इंडिया लिमिटेड
Nuclear Power Corporation of India Limited

The Nuclear Power Corporation of India Limited (NPCIL) is a premier public sector organization dedicated to harnessing nuclear energy for power generation in India. Established in 1987 under the Department of Atomic Energy, NPCIL has been instrumental in developing and operating nuclear power plants that significantly contribute to the nation's energy mix.

NPCIL's mandate encompasses the entire lifecycle of nuclear power plants—from design and construction to commissioning, operation, and maintenance. The organization manages a network of strategically located plants, including facilities at Tarapur, Narora, Kaiga, and Kakrapar, which are designed not only to meet the country's growing energy demands but also to adhere to the highest standards of safety and environmental responsibility.

A key strength of NPCIL is its rigorous commitment to nuclear safety and operational excellence. The corporation implements comprehensive safety protocols and continuous monitoring measures to ensure that its nuclear reactors operate under strict international standards. This focus on safety is complemented by ongoing investments in research and development, fostering technological innovation that improves reactor efficiency, enhances safety measures, and extends the operational life of its nuclear assets.

In addition to its technical achievements, NPCIL places significant emphasis on human resource development. The organization is renowned for its robust training programs, which are designed to cultivate a skilled and knowledgeable workforce capable of managing complex nuclear technologies. This commitment to excellence in talent development underpins NPCIL's reputation as a leader in the nuclear energy sector.

Moreover, NPCIL's efforts extend beyond power generation. It actively contributes to sustainable development by promoting clean energy solutions and adhering to environmentally friendly practices. Through its strategic initiatives and responsible corporate governance, NPCIL plays a pivotal role in ensuring energy security for India while paving the way for future innovations in nuclear technology.

Overall, NPCIL stands as a symbol of technological progress and responsible energy production, consistently driving India's journey toward a sustainable and secure energy future.

II. Internship Position

I interned as a Project Trainee at Nuclear Power Corporation of India Limited (NPCIL), where I gained hands-on experience by actively contributing to key projects. This role allowed me to collaborate with seasoned professionals, apply theoretical knowledge in real-world scenarios, and develop essential technical and problem-solving skills.

III. About Project

This project is centered around the development of an offline HR chatbot specifically designed to streamline the process of retrieving information from PDF-based HR notices. In many organizations, HR notices contain important information that employees often need to reference, whether it's related to company policies, benefits, schedules, or other essential updates. The aim of this project is to build a chatbot capable of efficiently extracting relevant information from these documents, enabling employees to access the data they need quickly and accurately.

The HR chatbot functions entirely offline, ensuring that it can be used in environments with limited or no internet connectivity, which is particularly beneficial for organizations with strict data security requirements. By leveraging natural language processing (NLP) and machine learning technologies, the chatbot understands user queries, processes them, and retrieves the appropriate information from stored PDF files. This solution is designed to automate the time-consuming task of manually searching through documents, thus saving employees valuable time and increasing productivity.

Additionally, the chatbot is intended to be user-friendly, offering an intuitive interface that simplifies the interaction for employees. It aims to provide precise and contextually relevant responses, enhancing the overall user experience. The project focuses on developing a solution that balances accuracy, speed, and computational efficiency, especially in environments with limited resources, making it both effective and resource-conscious. Ultimately, the goal of this project is to create a powerful tool that improves workplace efficiency by enabling employees to access important HR information effortlessly and on-demand.

Project Overview

The HR Chatbot project is designed to automate the extraction of relevant information from PDF-based HR notices in an offline environment. It aims to assist employees by quickly retrieving key details from organizational documents such as policies, announcements, and schedules, without requiring internet access. By using natural language processing (NLP), the chatbot interprets user queries and locates the relevant information within stored documents, offering an efficient and time-saving solution for HR-related inquiries.

The project involves extensive testing with various pre-trained language models like GPT-2, GPT-Neo, LLaMA, and Phi3 to balance response accuracy and inference speed within a CPU-only environment. Despite challenges such as high inference time and limited resources, the system was optimized for fast processing, delivering relevant and accurate responses. This solution enhances productivity by providing employees with quick, secure access to essential HR information, all while functioning offline and efficiently managing available resources.

Key Features:

- Project Goal: Develop an offline HR chatbot to extract data from PDF-based HR notices.
- Functionality: Provides quick, relevant information to employees without requiring internet access.
- Technology: Uses natural language processing (NLP) to understand queries and retrieve answers from stored documents.
- Pre-trained Models: Tested models such as GPT-2, GPT-Neo, LLaMA, and Phi3 for response accuracy and inference speed.
- Selected Model: LLaMA 3.2 1B was chosen due to its balance between inference speed and response accuracy.
- Reason for Selection: LLaMA 3.2 1B provided a significant reduction in response time (under 10 seconds) compared to larger models, making it efficient for the limited hardware resources available.
- Hardware Constraints: Optimized the system to run in a CPU-only environment without GPU support.
- Optimization: Focused on reducing inference time and ensuring contextual accuracy of responses.
- Outcome: Aimed to improve efficiency and productivity by enabling quick, secure access to HR-related information.

Technology Stack:

1. **Backend**:

- RAG (Retrieval-Augmented Generation): For improving response accuracy by combining document retrieval with generative capabilities.
- LangChain: Framework used to implement the RAG pipeline and manage interactions between the model and external data.
- Python 3.12: The programming language used for backend implementation.
- all-MiniLM-L6-v2: Embedding model used for efficient text representation and document retrieval.
- PDFPlumber: Package used to extract text from PDF documents.
- SpaCy Blank Model: Utilized for fine-tuning and processing user queries.

2. **Frontend**:

- HTML: Used for structuring the web interface.
- CSS: Used for styling and designing the user interface.
- Flask: Web framework that connects the backend with the frontend, enabling user interactions with the chatbot through a web interface.

Development Process:

1. Initial Research and Exploration
   - Model Selection: Started by researching various pre-trained language models such as GPT-2, GPT-Neo, LLaMA, and Phi3.
   - Objective: The goal was to find a model suitable for offline use that could efficiently retrieve and generate answers from HR notices in PDF format.
   - Challenges: Inference time and hardware limitations (CPU-only, no GPU) were identified as potential obstacles.

2. Model Fine-Tuning and Experimentation
   - Initial Attempts: Fine-tuned models like GPT-2 and GPT-Neo, but the results were not satisfactory in terms of accuracy and inference speed.
   - Shift to LLaMA: Moved to the LLaMA model, specifically LLaMA 3.2 8B, for better performance, but faced high inference times (around 600 seconds).
   - Optimization: Tested the LLaMA 3.2 1B model, which provided a significant improvement, reducing the inference time to under 10 seconds.
   - Model Comparison: Simultaneously experimented with the Phi3 model, which offered a faster response time of around 50 seconds, but was ultimately outperformed by the LLaMA 3.2 1B model.
3. Server Deployment and Resource Management
   - Server Room Deployment: Moved the project to a server with 768 GB RAM, but faced challenges with CPU utilization (100% usage for just one query).
   - Inference Optimization: Focused on optimizing the model for better performance, but resource limitations remained a key constraint, slowing down the process.
4. Frontend and Web Interface Setup
   - HTML/CSS: Designed the frontend of the chatbot using basic HTML for structure and CSS for styling.
   - Flask Integration: Set up a Flask web framework to integrate the frontend with the backend, allowing the chatbot to interact with users via a web interface.
5. PDF Text Extraction and Query Processing
   - PDF Extraction: Used PDFPlumber to extract textual data from PDF-based HR notices, enabling the chatbot to query the documents.
   - Query Handling: Utilized LangChain to implement the Retrieval-Augmented Generation (RAG) technique, ensuring the chatbot retrieves relevant information from the PDF and generates responses based on the user query.
   - Embedding Model: Implemented the all-MiniLM-L6-v2 embedding model to handle the text representation and document retrieval, enhancing the accuracy of the responses.

6. Fine-Tuning the Model for HR Queries
    - SpaCy Blank Model: Attempted fine-tuning using a blank model in SpaCy, but this approach faced challenges as it required exact match queries from the dataset, which was inefficient.
    - Re-evaluation: Shifted focus to fine-tuning existing models with fewer epochs to improve response relevance, though issues like gibberish responses and long processing times persisted.
7. Testing and Optimization
    - Testing: Ran multiple test cases to evaluate response time and relevance. Despite the improvements in inference speed, some out-of-context responses were still observed.
    - Optimization: Continued experimenting with prompt engineering, adjusting the model's prompts to improve response quality, but faced challenges with accuracy due to lack of GPU support for multiprocessing.
8. Feedback and Iteration
    - Demo and Feedback: Held a demo with IT team members, who suggested either building a model from scratch or further fine-tuning existing models.
    - Consideration for Future: Realized that building a model from scratch or fine-tuning it further would require more resources, time, and computational power.
9. Project Conclusion
    - Reflection: Overcame several technical hurdles, such as resource limitations and hardware constraints, while learning about model optimization and deployment in restricted environments.
    - Final Outcome: The project demonstrated the potential of developing an efficient, offline HR chatbot, though it highlighted the challenges of working without GPUs and with limited resources.

IV. Implementation.

1. Extract Text from PDFs (extract_text_from_pdf):
- This function opens the PDF using pdfplumber, extracts text from each page, and then performs multiple cleaning operations (like removing HTML tags, special characters, URLs, and newlines) using regular expressions.
- The cleaned-up text is returned.

2. Load Text from Folder (load_text_from_folder):
- This function loops through the files in the provided folder path.
- If a file is a PDF, it calls extract_text_from_pdf to extract and clean the text from that PDF.
- It combines the extracted text from all PDFs and returns it as a single string.

3. Chunk Text (chunk_text):

- This function uses RecursiveCharacterTextSplitter to split the extracted text into smaller chunks.
- The chunk size is defined by chunk_size, and the overlap between chunks is controlled by chunk_overlap.

4. Create Embeddings (create_embeddings):

- This function initializes the HuggingFace embedding model (all-MiniLM-L6-v2) and creates embeddings for each of the text chunks.
- These embeddings are then stored in a FAISS vector store.

5. Prepare Database (prepare_database):

- It first checks if a FAISS database already exists.
- If the database exists, it skips the preparation process.
- If the database doesn't exist, it loads PDFs, extracts the text, chunks the text, creates embeddings, and stores them in a FAISS vector store.
- After processing the text, the FAISS database is saved for future use.

6. Load LLM Model (load_llm):

- This function loads the language model (LLM) from the given file path using LlamaCpp.
- It sets up the model's parameters like temperature, max tokens, and threads for efficient processing.

7. Create Retrieval-Augmented Generation (RAG) Chain (create_rag_chain):

- This function sets up the RAG chain using the RetrievalQA class from Langchain.
- It prepares a prompt template to ensure that the model only answers based on the context passed in.
- The RAG chain uses the FAISS vector store for retrieving context and the LLM for generating the final answer.

8. Initialize Chatbot (initialize_chatbot):

- This function initializes the chatbot by loading the embeddings and database.
- It checks if the embeddings and FAISS database exist locally. If not, it loads and stores them.
- It loads the LLM model and creates the RAG chain for answering queries.

9. Answer Query (answer_query):

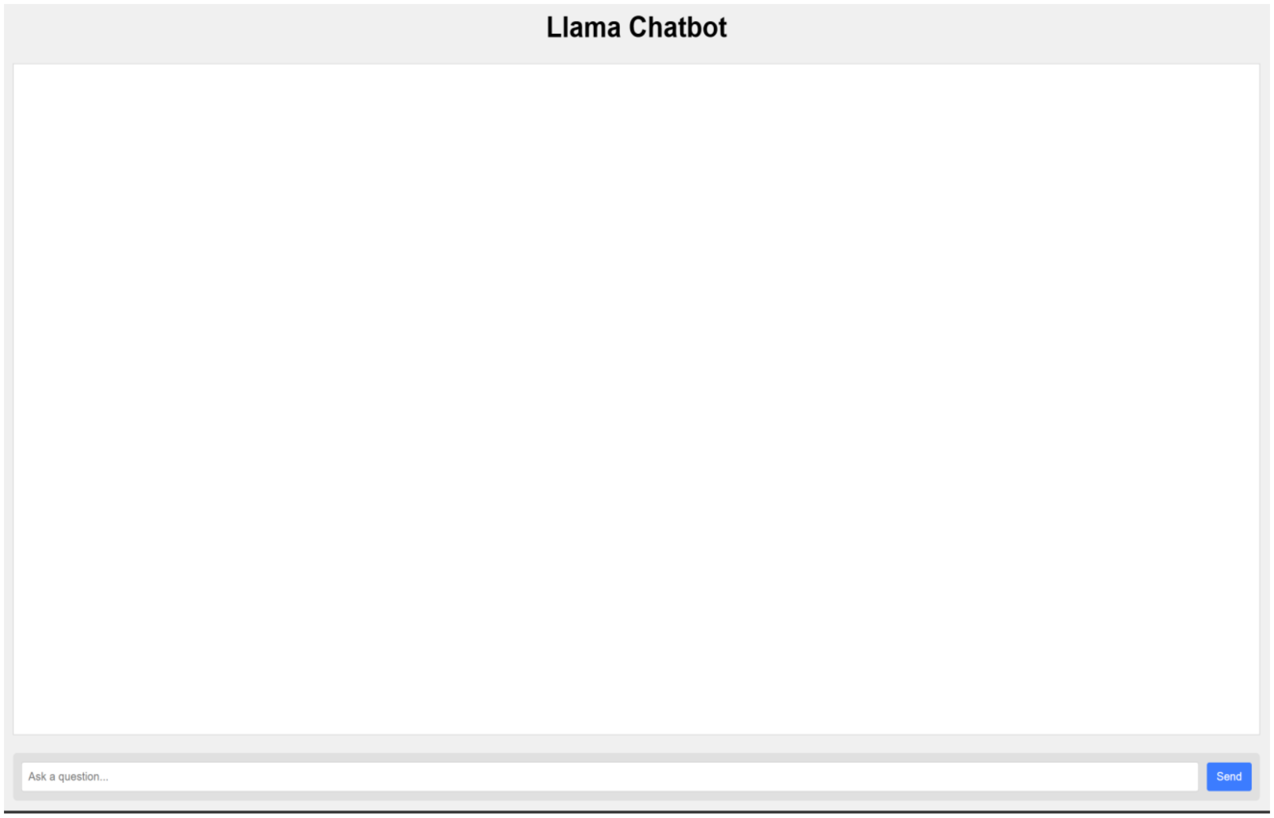
- This function handles answering a user query.
- If the query is a greeting like "hi" or "hello", it returns a predefined response.
- For other queries, it uses the RAG chain to retrieve relevant context from the FAISS database and generates a response using the LLM.
- The time taken to process the query is also measured.

10. Process Query (process_query):

- This function processes a single query asynchronously.
- It generates a unique query ID, processes the query using answer_query, and retrieves the response from the cache.
- The function then returns the response, time taken, and query ID.

11. Main Function (main):

- The main function initializes the database and chatbot.
- It continuously prompts the user for input (queries) and calls process_query to generate answers.
- The results are printed, including the answer, time taken to process, and query ID for tracking.



(Llama Chatbot User Interface)

V. Internship Description

During my internship at NPCIL, I worked on the development of an offline HR chatbot aimed at extracting and processing information from PDF-based HR notices. The goal of the project was to create a solution that would allow employees to quickly access HR-related information without requiring internet access, making it ideal for secure environments.

I explored and experimented with various pre-trained language models, including GPT-2, GPT-Neo, LLaMA, and Phi3, to identify the best fit in terms of response accuracy and inference speed. The project involved using LangChain for implementing Retrieval-Augmented Generation (RAG) to improve the chatbot's responses, and I utilized the all-MiniLM-L6-v2 embedding model for efficient document retrieval. The backend was developed in Python 3.12, with a simple web interface built using HTML, CSS, and the Flask framework.

Throughout my internship, I faced significant challenges due to resource limitations, such as working in a CPU-only environment without GPU support, which affected inference speed and model performance. Although the project is still ongoing, I have gained valuable experience in machine learning model optimization, natural language processing, and deploying AI systems in constrained environments. This project has provided me with insights into handling technical limitations and working toward efficient solutions in real-world scenarios.

VI. Challenges Faced

1. Resource Limitations:

The project was implemented in an environment with CPU-only resources, and the system had no access to a GPU. This resulted in significantly high inference times, especially when working with larger models like LLaMA 3.2 8B, which affected both performance and efficiency.

2. High Inference Time:

The initial models tested, including LLaMA 3.2 8B, resulted in very slow inference times, with some models taking up to 600 seconds per query. Although switching to smaller models like LLaMA 3.2 1B improved response times, it still required significant optimization to achieve satisfactory results.

3. Hardware Constraints:

The CPU utilization reached 100% for even a single query, limiting the ability to perform parallel processing. This made it impossible to run multiple queries concurrently, severely hindering scalability and leading to slow responses. Despite deploying the model on a high-performance server with 768 GB RAM, the lack of GPU support presented ongoing challenges in terms of computational power.

4. Package Installation Issues:

Due to the offline environment and security restrictions, installing required packages and dependencies became a significant challenge. Without internet access, installing new libraries and updating existing ones was complex and time-consuming, requiring manual installations and workarounds.

5. Data Preprocessing and Extraction:
Extracting relevant data from PDF notices was a complex task. Despite using tools like PDFPlumber to extract text, issues with inconsistent formatting and missing data in the PDFs led to additional challenges in ensuring accurate document retrieval and processing.

6. Fine-Tuning Challenges:
While fine-tuning models like SpaCy Blank Model, the requirement for exact match queries between the dataset and user input made the process inefficient. Generating relevant responses was difficult without significant model adjustments, and tuning the models with fewer epochs initially led to gibberish responses and long processing times.

7. Model Optimization:
Although RAG and LangChain were implemented to enhance retrieval and generation, optimizing for speed and contextual accuracy was a constant challenge due to the limited computational resources and the complexity of the tasks.

8. Deployment Issues:
Transitioning the project to the server room provided some speed improvements, but issues such as slow response times and out-of-context responses still persisted despite extensive adjustments to prompts and model fine-tuning.

These challenges provided valuable learning opportunities, allowing me to gain practical experience in overcoming technical limitations, managing resource constraints, and deploying AI models in offline environments.

VII. Future Scope

To enhance the performance of the HR chatbot, setting up a proper GPU would significantly improve inference speed and reduce processing time, especially for larger models. Currently, with CPU-only resources, the system faces challenges in handling high computational demands, leading to slow response times and 100% CPU utilization for even a single query. Integrating a GPU would allow for faster processing and more efficient model execution, enabling the system to handle multiple queries simultaneously. Additionally, the ability to implement parallel querying would significantly optimize the chatbot's performance, allowing it to manage multiple user requests concurrently without overloading the CPU. This would result in a more responsive and scalable system, capable of serving a larger number of users efficiently.

VIII. Summary

During my internship at NPCIL, from December 9, 2024, to March 31, 2025, I worked on developing an offline HR chatbot designed to extract and retrieve data from PDF files, specifically HR-related notices. The process involved several stages, including text extraction from PDF documents using pdfplumber, cleaning the extracted data, and chunking the text into manageable pieces. I then created embeddings for these text chunks using the all-MiniLM-L6-v2 model and stored them in a FAISS vector database. The chatbot was built using a Retrieval-Augmented Generation (RAG) framework that incorporated the LlamaCpp language model to generate responses based on the embedded data. This offline solution was designed to function without internet access, ensuring the organization could retrieve information from internal documents securely. The project focused on handling user queries and providing accurate, context-based responses by leveraging the power of embeddings and large language models, all while operating within the organization's limited resources.

IX. References

- PyPI: https://pypi.org/ for package installations and versioning.
- Flask Documentation: https://flask.palletsprojects.com/ for Flask web framework details.
- Hugging Face Models: https://huggingface.co/models for accessing pre-trained models.
- Meta LLaMA: https://ai.facebook.com/blog/large-language-model-llama/ for information on LLaMA models.
- FAISS: https://github.com/facebookresearch/faiss for vector database and similarity search.
- GitHub: https://github.com/ for open-source code repositories and community contributions.
- Stack Overflow: https://stackoverflow.com/ for development-related discussions and solutions.
- Reddit: https://www.reddit.com/r/MachineLearning/ for machine learning discussions.
- Quora: https://www.quora.com/ for general technical and AI-related discussions.