Problem Statement

The Problem Statement i.e. “**Running GenAI on Intel AI Laptops and Simple LLM Inference on CPU and fine-tuning of LLM Models using Intel® OpenVINO™**” is designed to introduce beginners to the exciting field of Generative Artificial Intelligence (GenAI) through a series of hands-on exercises.

The primary objectives are:

- Implement a GenAI application capable of running on Intel AI Laptops.
- Perform efficient Large Language Model (LLM) inference on CPU.
- Explore the process of fine-tuning an LLM model to create a custom chatbot.
- Demonstrate practical application of LLM models in document processing and question-answering.

Unique Idea Brief (Solution)

Our solution is an **intelligent PDF chatbot** that allows users to **upload PDF documents** and **ask questions** about their content in **natural language**.

This approach addresses the major challenges outlined in the problem statement:

1. Handling large pre-trained language models with significant file sizes
2. Performing LLM inference on CPU efficiently
3. Demonstrating the concept of fine-tuning (through an alternative approach)
4. Creating a custom Chatbot using Intel AI Tools

Key aspects of our approach include:

- Semantic chunking for context preservation
- Hybrid retrieval-generation approach for accurate and creative responses
- CPU-optimized inference using Intel's OpenVINO toolkit
- Adaptive prompt engineering for improved response relevance
- Scalable vector storage with FAISS for efficient similarity search

Unique Idea Brief (Solution)

Now, instead of implementing traditional fine-tuning, in this project we focused on **leveraging effective retrieval mechanisms** and **advanced prompt engineering, offering:**

- **Generalizability** across various PDF contents
- **Resource efficiency**
- **Ease of maintenance** and **rapid adaptation**
- Preserved **model generality**

Development Process Overview:

1. **Environment Setup**: Used **Google Colab**, resolved library compatibility issues by specifying exact versions for critical dependencies.
2. **PDF Processing**: Implemented **text extraction** using **PyPDF2**, focused on text-based PDFs.

Unique Idea Brief (Solution)

3. **Semantic Chunking**: Developed custom function for **context preserving** text division to balance chunk size and overlap for **optimal context preservation**.
4. **Embedding and Vector Store**: Utilised **SentenceTransformer** for embeddings and **FAISS** for Vector Storage. Choosing the right embedding model was critical, after experimenting a lot, we selected **'all-MiniLM-L6-v2'** for its balanced performance and resource efficiency.
5. **Language Model**: Integrated **TinyLlama**, optimized with **OpenVINO** to achieve efficient **inference on CPU**.
6. **Chatbot Function**: Created **query processing** and **response generation** system, ensuring response relevance to specific PDF content by refining the prompt structure to improve answer quality.
7. **User Interface**: Built **intuitive interface** using **Gradio** to ensure a smooth user experience while handling potentially **long-running operations**(like PDF processing) required careful consideration of **asynchronous processing** and **user feedback mechanisms.**

Features Offered

1) **PDF Upload and Processing**
    - Supports various PDF formats and sizes
    - Efficiently extracts text while preserving document structure
    - Handles multi-page documents seamlessly
2) **Natural Language Querying**
    - Accepts questions in free-form natural language
    - Understands context and intent of user queries
    - Supports follow-up questions and maintains conversation context
3) **Intelligent Information Retrieval**
    - Utilizes semantic search for high-accuracy information retrieval
    - Implements FAISS for fast and efficient similarity search
    - Ranks and selects the most relevant text chunks for each query
4) **AI-Powered Responses**
    - Generates human-like responses using the TinyLlama language model
    - Ensures responses are contextually relevant to the PDF content
    - Provides citations or references to specific parts of the PDF when appropriate

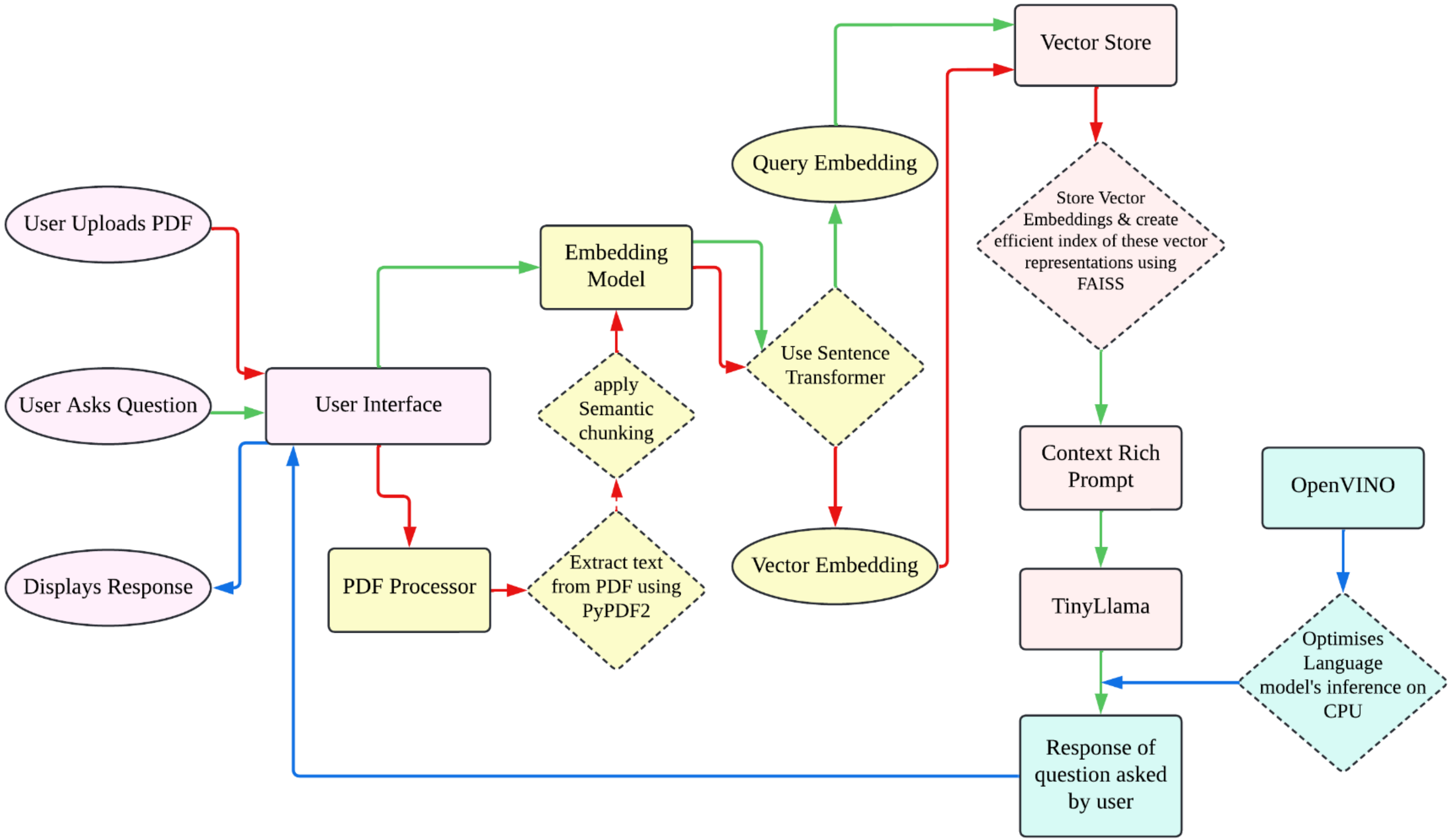
Features Offered

5) **CPU Optimization with Intel OpenVINO**
   - Leverages Intel's OpenVINO toolkit for optimized inference on CPU
   - Enables efficient running of large language models on standard laptops
   - Reduces inference time and resource usage compared to non-optimized implementations
6) **User-Friendly Gradio Interface**
   - Intuitive design for easy PDF upload and question input
   - Real-time response generation with visible processing status
   - Option to restart the chatbot or upload a new PDF without refreshing the page
7) **Semantic Chunking**
   - Divides PDF text into meaningful semantic chunks
   - Preserves context within chunks for more accurate information retrieval
   - Allows for processing of long documents while maintaining semantic coherence

Process flow

1. **PDF Upload and Processing**:
    - User uploads a PDF file
    - System extracts text from the PDF
    - Text is divided into semantic chunks
    - Chunks are converted into embeddings and stored in a vector database
2. **Query Processing**:
    - User inputs a question
    - Question is converted to an embedding
    - System retrieves most relevant chunks using similarity search
3. **Response Generation**:
    - Retrieved chunks are used to create a context-rich prompt
    - Prompt is fed into the LLM for response generation
    - Generated response is presented to the user

Architecture Diagram



Technologies used

- **Python**: Primary programming language
- **PyPDF2**: For PDF text extraction
- **Sentence Transformers**: For generating text embeddings
- **FAISS**: For efficient similarity search
- **Hugging Face Transformers**: For accessing pre-trained language models
- **Intel OpenVINO**: For optimized model inference on CPU
- **NLTK**: For text processing tasks
- **Gradio**: For creating the user interface
- **Google Colab**: Development and deployment platform

Team members and contribution:

- **Swetakshi Nanda**: Project lead, architecture design
- **Pratyush Pahari**: LLM integration and OpenVINO optimization.
- **Arpan Bag:** PDF processing, Embedding generation
- **Akashdeep Mitra**: User interface development and integration.
- **Tulika Chakraborty**: Documentation of the complete Project.

Conclusion

Our PDF Chatbot project demonstrates the practical application of Generative AI techniques in **information retrieval and question answering**, aligning with the objectives set forth in the Problem Statement. By leveraging advanced NLP models and optimizing for CPU inference, we've created **a system that allows users to efficiently extract insights from PDF documents through natural language interaction.**

Key achievements:

- Successful implementation of a GenAI application optimized for Intel AI Laptops
- Efficient LLM inference on CPU using OpenVINO
- Creation of a user-friendly interface for PDF querying
- Implementation of a hybrid retrieval-generation approach for accurate responses, addressing the challenge of customizing LLMs without traditional fine-tuning

Our approach of combining efficient retrieval, dynamic prompt engineering, and optimized inference achieves the goals of accuracy and relevance while maintaining flexibility and ease of use. This strategy aligns well with the project's objectives of demonstrating practical GenAI applications and efficient LLM inference on CPU.

Future improvements for the PDF Chatbot project focus on enhancing its capabilities and performance. These include comparing traditional fine-tuning with the current approach, expanding document type support, optimizing long document processing, and integrating few-shot learning techniques to improve response quality for specific question types.