Hybrid QA System (BERT + RAG)

Author: Group 9

Version: 1.0

Purpose: AI-powered Question Answering system with Manual, Batch, and Retrieval-Augmented (RAG) modes using BERT and Sentence-BERT

Framework: Python + Streamlit + Transformers + SentenceTransformers

# 🧠 Project Summary

This project is a modular Hybrid QA Assistant that supports multiple modes of operation:  
1. Manual QA: Ask questions over a given context  
2. Batch Evaluation: Run QA performance on SQuAD2.0-style datasets  
3. RAG QA: Upload documents (PDF, DOCX, TXT) and ask questions  
4. Settings: Centralized control panel for thresholding and model behavior

# 📂 Folder Structure

Hybrid\_QA\_App/  
├── app.py ← Main Streamlit entry point  
├── requirements.txt ← Python dependencies  
├── qa\_model\_checkpoint/ ← Fine-tuned BERT QA model directory  
├── data/ ← SQuAD2.0 datasets, uploaded documents  
├── tabs/ ← Streamlit tab modules  
│ ├── tab1\_manual\_qa.py ← Manual QA  
│ ├── tab2\_batch\_eval.py ← SQuAD2.0 Evaluation  
│ ├── tab3aq\_rag.py ← RAG QA from uploaded documents  
│ └── settings.py ← Central configuration panel  
├── utils/ ← Reusable helper functions  
│ ├── extract\_text.py ← PDF/DOCX/TXT extraction  
│ ├── rag\_pipeline.py ← Chunking, embedding, semantic search  
│ ├── squad\_eval.py ← EM, F1, semantic score metrics  
│ └── visualization.py ← Plots and charts

# 🚀 How to Run

1. Clone the repository:  
 git clone https://github.com/your-org/hybrid-qa-assistant.git  
 cd hybrid-qa-assistant

2. Install dependencies:  
 pip install -r requirements.txt

3. Run the Streamlit app:  
 streamlit run app.py

# ⚙️ Global Settings (Tab 4)

Includes:  
- No-answer threshold  
- Confidence threshold  
- Semantic similarity threshold  
- Enable semantic override  
- Max span length  
- Model selector

# 🧪 Tab Features

Tab 1: Manual QA  
- Input: Context + Question  
- Output: Answer + confidence + similarity  
- Suppression: Based on no-answer + semantic

Tab 2: Batch Evaluation  
- Input: SQuAD2.0 JSON/CSV  
- Output: EM, F1, no-answer accuracy  
- Visuals: Confusion matrix, histograms, line charts

Tab 3: RAG QA  
- Input: Uploaded document  
- Process: Chunk → Embed → Retrieve → QA  
- Output: Best chunk + extracted answer + confidence

Tab 4: Settings  
- Central control panel synced via session\_state

# 🔍 Dependencies

transformers, sentence-transformers, streamlit, PyPDF2, docx2txt, matplotlib, plotly, seaborn, pandas, torch

# 🔮 Future Extensions

Add retraining interface, multilingual support, or Local ChatGPT integration.

# 📌 Notes

Ensure your BERT model folder contains HuggingFace-compatible files.

# 👥 Team Instructions

Each team member can work on a tab independently. Use `utils/` for shared logic.

# 🧾 License & Acknowledgements

This project is for Capstone project for submitting on the final day in MIIT use and references open-source tools including HuggingFace Transformers and SQuAD 2.0.