Conversational Interface for Multi-Source Information Retrieval

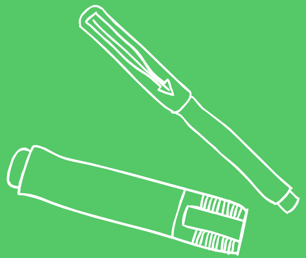
Abhiyan Sainju

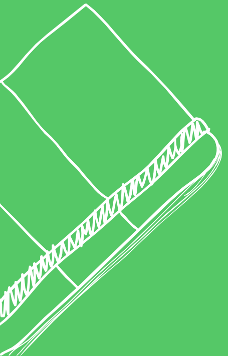
Balaji Senthilkumar

Introduction

- Challenges of information overload from multiple sources
- Inefficiency of manual searching and filtering
- Results in frustration and lost productivity
- Underutilization of audio content due to lack of efficient tools
- Our project aims to streamline information retrieval process

Project Overview





Project Overview

- Conversational interface for querying multiple sources of information including PDFs, YouTube videos, and audio recordings
- Leverages NLP, vector databases, large language models
- Allows users to interact with information efficiently using context memorization

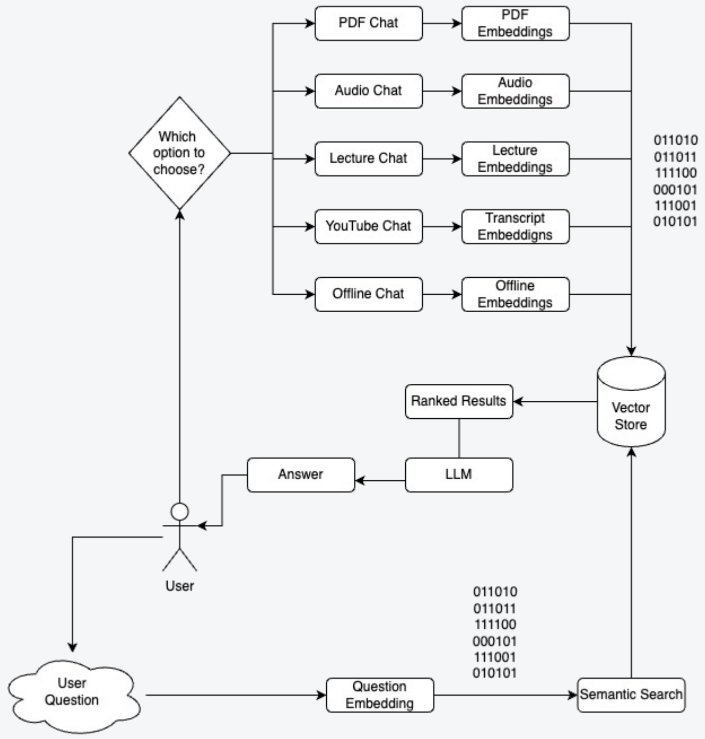
Core Features

- Text extraction from PDFs, YouTube transcripts, audio files
- Vector embedding and storage in vector databases
- Conversational query interface to interact with the information present in PDFs, YouTube videos and audio files.
- Integration of online (GPT-3.5-turbo) and offline (Mistral) language models

System Architecture

- User options: PDF Chat, Audio Chat, Lecture Chat, YouTube Chat, Offline Chat
- Files processed to extract text/transcripts and convert to embeddings
- Embeddings stored in respective vector databases (FAISS or Chroma)
- User enters query in conversational chat interface
- Query embedded and matched to similar embeddings in vector store
- Top result sent to language model (GPT-3.5-turbo or Mistral) to generate response
- Response displayed in chat, maintaining conversation history/context

How our System Works



Implementation Details

- Text extraction using PyPDF2, chunking with CharacterTextSplitter
- Embeddings with OpenAIEmbeddings (online), OllamaEmbeddings (offline)
- Vector stores: FAISS (online), Chroma (offline)
- Conversational interface using ConversationalRetrievalChain, ConversationBufferMemory components

Youtube Video Processing

- Use youtube-transcript-api to fetch video transcript
- Extract video id from YouTube link using regex
- Process transcript text like PDF files
- Generate embeddings and store in vector database
- Allow querying video content via chat

Audio Processing

- Record audio from user via browser
- Convert to byte streams, store as temporary file
- Use OpenAI's whisper-1 model for speech-to-text
- Process transcript to create embeddings/vector store
- Enable querying audio through chat interface

Offline LLM Integration

- Use Mistral offline language model
- Local processing, addresses privacy concerns
- OllamaEmbeddings for local embedding generation
- Chroma as offline vector store

Difficulties Faced

- Maintaining session state for context memory, preventing conflicts
- Dealing with deprecated libraries/code
- Improving inference speed
- Enabling GPU acceleration for offline model

Contributions and Achievements

- Efficient multi-source information retrieval
- Online and offline language model support
- Contextual and conversational interactions
- User-friendly interface with Streamlit
- Time-saving and productivity enhancements

Future Work

- Expanded source support (websites, databases, knowledge bases)
- Multilingual audio support
- Personalized query suggestions and result ranking
- GPU acceleration for offline models

Thanks!

Any questions?

Credits

- Presentation template by SlidesCarnival
- Photographs by Unsplash
- Illustrations by Undraw.co