PROJECT DOCUMENTATION

Project Title: Automated Podcast Transcription and Topic Segmentation

Project Type: End-to-End AI/ML Pipeline & Web Application

Tech Stack: Python, OpenAI Whisper, BERT, Flask, Plotly

1. Abstract

The rapid growth of digital audio content, particularly podcasts and virtual meetings, has created a significant challenge in information retrieval. Unlike text, audio is linear and difficult to search. This project presents an automated system designed to transform unstructured audio files into structured, navigable knowledge bases. By integrating state-of-the-art Deep Learning models—specifically **OpenAI Whisper** for Automatic Speech Recognition (ASR) and **BERT** for semantic analysis—the system generates accurate transcripts, detects topic shifts without human intervention, and creates concise summaries. The final output is presented via an interactive Web Dashboard, allowing users to visually navigate audio content, search for specific keywords, and analyze sentiment trends in real-time.

2. Introduction

2.1 Problem Statement

Listening to long-form audio (1–3 hours) to find specific information is inefficient. Traditional media players lack semantic navigation, meaning users must scrub through timelines blindly. Manual transcription and summarization are prohibitively expensive and slow for large archives.

2.2 Proposed Solution

We propose an automated "Podcast Intelligence" pipeline that:

1. **Transcribes** speech to text with high accuracy.
2. **Segments** the text into distinct chapters based on topic changes (not just silence).
3. **Summarizes** each chapter to provide a "Table of Contents" for the audio.
4. **Visualizes** the conversation's sentiment and keywords.

2.3 Objectives

- Develop a robust preprocessing pipeline for real-world noisy audio.

- Implement semantic segmentation using embedding similarity (Sentence-Transformers).
- Create a user-friendly Web UI for non-technical users to upload and analyze files.

3. System Analysis

3.1 Functional Requirements

- **Input:** The system must accept .mp3 and .wav formats up to 2GB.
- **Processing:** It must handle multi-speaker dialogue and normalize audio volume.
- **Output:** It must generate a JSON file containing the transcript, segmented topics, summaries, and sentiment scores.
- **Interface:** Users must be able to click a topic to seek the audio player to that exact timestamp.

3.2 Software & Hardware Requirements

- **Operating System:** Windows 10/11, Linux, or macOS.
- **Programming Language:** Python 3.9+.
- **Core Libraries:** torch (PyTorch), transformers (Hugging Face), flask (Web Framework), scikit-learn.
- **External Tools:** FFmpeg (for audio codec conversion).
- **Hardware:** Minimum 8GB RAM. (GPU recommended for faster inference but runs on CPU).

4. System Architecture and Flow

4.1 Data Flow Diagram

The system follows a sequential pipeline architecture:

1. **Data Ingestion:**
    - User uploads audio via the Web Dashboard.
    - System validates the file format and saves it to a temporary directory.
2. **Preprocessing Layer (preprocessing):**
    - **Normalization:** Audio volume is leveled (standardized DB) using pydub.
    - **Conversion:** File is converted to 16kHz Mono WAV (optimal for ASR).

3. **Transcription Layer (models):**
    - **Model:** OpenAI Whisper (Tiny/Base).
    - **Process:** Audio is processed in 30-second windows using an Encoder-Decoder Transformer architecture.
    - **Output:** Raw text with start/end timestamps.
4. **NLP Core Layer (segmentation & summarization):**
    - **Segmentation:** Sentences are converted into vector embeddings using **MiniLM-L6-v2 (BERT)**. The system calculates Cosine Similarity between adjacent sentences. A sharp drop in similarity triggers a new "Topic Segment."
    - **Enrichment:**
        - **Summarization:** distilbart-cnn-12-6 generates 2-sentence abstracts.
        - **Keywords:** KeyBERT extracts context-aware tags.
        - **Sentiment:** TextBlob assigns polarity scores (-1 to +1).
5. **Presentation Layer (web_app):**
    - **Flask Server:** Routes data to the frontend.
    - **Frontend:** HTML5/CSS3 dashboard with Plotly.js charts for visual analytics.

5. Project Directory

The project is structured to separate source code, data, and logs for maintainability.

Plaintext

```
Automated-Podcast-Transcription/
├── data/
│   ├── transcripts/        # Intermediate JSON outputs
│   ├── final_output/       # Processed Analysis Data (JSON)
│   └── temp_upload/        # Temporary raw uploads
├── logs/
│   └── app.log             # System activity and error logs
├── src/
```

```
|   ├── preprocessing/
|   |   └── data_loader.py    # Audio normalization logic
|   ├── models/
|   |   └── transcriber.py    # Whisper ASR wrapper
|   ├── segmentation/
|   |   └── semantic_segmenter.py # BERT-based topic splitter
|   ├── summarization/
|   |   └── content_processor.py  # Summaries, Keywords, Sentiment
|   ├── utils/
|   |   └── logger.py         # Logging configuration
|   └── web_app/
|       ├── app.py            # Main Flask Application
|       └── templates/
|           ├── index.html    # Dashboard Interface
|           └── player.html   # Detailed Analysis Interface
├── .env                      # Environment Variables
├── requirements.txt          # Dependency list
└── README.md                 # Documentation
```

6. Core Methodology

6.1 Semantic Segmentation Algorithm

Unlike standard approaches that split audio based on silence (pauses), this project uses **Semantic Segmentation**.

1. **Embeddings:** Every sentence in the transcript is converted into a 384-dimensional dense vector using a pre-trained BERT model.
2. **Similarity Matrix:** We calculate the cosine similarity between sentence i and sentence $i+1$.

3. **Boundary Detection:** If the similarity score drops below a dynamic threshold (e.g., 0.5), it indicates a shift in meaning, marking a new segment.

6.2 Keyword Extraction (KeyBERT)

We utilize **KeyBERT** to extract keywords that are representative of the entire segment. It uses BERT embeddings to find n-grams that are most similar to the document itself, ensuring keywords capture the *context*, not just frequency.

7. Result and Performance

7.1 Performance Metrics

- **Transcription Accuracy:** The tiny model achieves ~80% accuracy on clear English audio. The base model improves this to ~90-95%.
- **Processing Speed:**
    - Audio Preprocessing: < 5 seconds.
    - Transcription (Tiny): ~10% of audio duration (e.g., 6 mins for 1-hour audio).
    - Segmentation & NLP: < 30 seconds for 1 hour of text.

7.2 Visual Analytics

The User Interface successfully renders:

- **Sentiment Timeline:** A spline chart showing emotional trajectory over time.
- **Keyword Cloud:** A time-based bubble chart clustering key terms by their occurrence.

8. User Manual

8.1 Installation

1. **Clone/Download** the repository.
2. **Create Environment:** python -m venv venv
3. **Activate:** venv\Scripts\activate (Windows) or source venv/bin/activate (Mac/Linux).
4. **Install Dependencies:** pip install -r requirements.txt
5. **Setup FFmpeg:** Ensure FFmpeg is installed and added to the system PATH.

8.2 Configuration

Create a .env file in the root directory:

Ini, TOML

AUDIO_DIR=path/to/your/audio/folder

SECRET_KEY=your_secret_key

8.3 Execution

1. Run the web server:

Bash

python src/web_app/app.py

2. Open browser to: http://127.0.0.1:5000/
3. Upload an audio file or browse existing processed episodes.

9. Impact and Future Scope

9.1 Impact

This tool democratizes access to audio data. It allows researchers, students, and professionals to "read" audio, saving hours of manual review time. In clinical or legal settings, it ensures critical details are indexed and searchable.

9.2 Future Scope

- **Speaker Diarization:** Identify *who* is speaking (e.g., "Doctor" vs "Patient") using PyAnnotate.
- **Multilingual Support:** Upgrade Whisper to support translation for non-English podcasts.
- **Cloud Deployment:** Dockerize the application for scalable deployment on AWS or Google Cloud.
- **RAG (Retrieval-Augmented Generation):** Add a Chatbot interface allowing users to "Ask questions" to the podcast.