Multimodal Text Processing System: An Integrated Approach

Alp Yalcinkaya
yalcinkaya.a@northeastern.edu
MSCS-Align Northeastern University

Heet Manish Kanani
kanani.h@northeastern.edu
MSCS-Northeastern University

Purva Sanjay Agarwal
agarwal.pu@northeastern.edu
MSCS-Northeastern University

Utkarsh Gogna
gogna.u@northeastern.edu
MSCS-Northeastern University

Yi Zu
zu.y@northeastern.edu
MSCS-Align Northeastern University

***Abstract*—This project develops a unified multimodal text processing system that integrates speech-to-text transcription, text summarization, and named entity recognition using advanced models like Whisper, BART, and a custom BiLSTM-CRF. The system effectively converts raw audio into structured, actionable text data, demonstrating robust performance and offering potential applications in automated transcription, content summarization, and data analysis.**

***Index Terms*—NER, BiLSTM-CRF, WER, BART**

I. Introduction And Background

A. Introduction

In the modern data-driven world, the rapid proliferation of multimedia content essence, heterogeneous in nature, with speech, videos, and text being the most common forms, has increased the development of systems that can process and extract useful insights from these modalities. The seamless conversion of spoken language to structured text and key entities is an important requirement of applications like automated transcription, meeting summarization, and data analysis. However, current solutions address these tasks in a rather fragmented way, which, in turn, results in inefficiencies and gaps in the processing pipeline.

B. Motivation

This calls for systems that can handle both speech and text with equal ease. Though there existing solutions, either they excel in a single text-transcription, summarization and classification or they can't maintain seamless integration across the whole pipeline. This work tries to fill this gap by implementing transcription, summarization, and classification within a single robust system.

C. Contextualisation within Literature

Each of these components of the proposed system has seen considerable advances:

- Whisper is an autoregressive speech-to-text model outperforming its competitors on many multilingual transcription tasks. It has achieved robust performances even in noisy conditions. Zero-shot capabilities make it a very desirable system to use on diverse audio inputs with limited domain-specific fine-tuning. Other competitive solutions, like those offered from Google's ASR, also provide competitive performance but may fall short when it comes to Whisper's versatility in zero-shot scenarios [1].
- Text Summarization: Abstractive summarization has evolved, especially based on transformer-based architectures like BART. Considering a denoising sequence-to-sequence framework, the generated summaries from BART preserve both fluency and conciseness [4]. Previous works already discussed its high performances regarding summarization of highly abstractive datasets like XSum and already confirmed strong results with higher ROUGE and BERTScore metrics compared to state-of-the-art systems.
- Named Entity Recognition: BiLSTM-CRF still represents a foundational approach for NER, putting in leverage from bidirectional contextual embeddings. from LSTMs through the CRF layer's enforcement of consistent tagging. across sequences. While newer transformer-based models like BERT have emerged, this architecture remains very efficient for domain-specific tasks with smaller datasets. While there has been some progress in each of these areas separately, relatively few efforts have been made to integrate these into one framework. Previous works focus on optimizations within each task separately while not addressing cascading issues. Errors and interdependencies in a multimodal pipeline. This work closes this gap by presenting an integrated system processing audio data end-to-end from transcription over summarization to entity extraction.

II. Methods

A. System Architecture

Our multimodal text processing system integrates three major components in a sequential pipeline: Speech-to-Text transcription, Text Summarization, and Named Entity Recognition. The pipeline is orchestrated through a main script that executes each component in sequence, with each module implemented as a standalone Python script. The system implements basic error handling and validation through file existence checks and subprocess execution monitoring. Each processing stage verifies its outputs before proceeding to the next step, with error reporting for failed executions. The pipeline maintains a simple directory structure for data flow, with each component reading from and writing to specified locations within a demo directory. Individual summarized texts are concatenated into a single combined file before being passed to the Named Entity Recognition stage, ensuring unified processing of all summarized content. The pipeline's execution flow is tracked through console outputs that provide status updates and completion confirmations for each stage. Error handling includes catching and reporting both subprocess execution failures and general exceptions, enabling identification of which pipeline stage encountered issues. The system tracks basic performance metrics such as total execution time and maintains counts of processed files at each stage.

1) ***Speech to Text:*** The speech-to-text module utilizes OpenAI's Whisper [3] base model for audio transcription, including several optimizations for processing long-form audio. The system employs a chunking strategy that splits the audio into 30-second segments with 2-second overlaps to maintain transcription continuity. This overlap duration was chosen to capture complete phrases (4-6 words) at chunk boundaries, accounting for natural English speaking rates of 2-3 words per second and typical speech patterns including sentence transitions and pauses. The speech-to-text module uses Whisper's pre-trained model with specific configuration settings to optimize transcription quality. The system uses beam search (a technique that explores multiple possible transcription paths simultaneously to find the best result) with 5 parallel beams, while preventing repetitive phrases through n-gram control. The preprocessing involves two main steps: first converting audio into visual representations of sound frequencies (spectrograms), then transforming these into tokens the model can process. The module is configured specifically for English language output and includes features to prevent common transcription issues like repeated phrases or words. The system handles long-form audio effectively using the TEDLIUM dataset, which consists of approximately 11.5-minute samples on average. The chunking implementation effectively handles extended audio while maintaining transcription quality, achieving an average Word Error Rate (WER) of 0.3082 across the dataset.

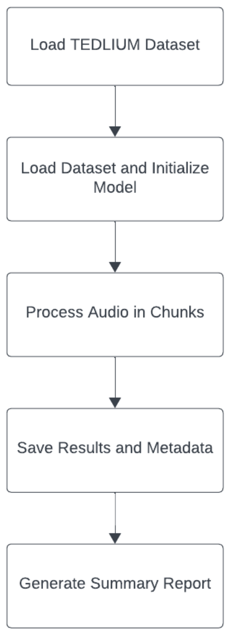

Fig. 1. Speech to text pipeline

*2) **Text Summarization**:* The summarization module implements a two-pass architecture using the BART-large-CNN model [2]. The system processes long-form content through a chunking mechanism that splits text into manageable segments of 900 tokens maximum, implementing sentence-aware boundaries to maintain semantic coherence. This conservative token limit, reduced from the model's 1024 maximum, provides a safety buffer to prevent index overflow errors during processing.

The basic summarization pipeline uses beam search with 4 parallel beams and a length penalty of 2.0. During the first pass, it will process each chunk independently. When the concatenated intermediate summaries pass a certain threshold of tokens, it will automatically trigger a second pass summarizing the concatenation of said outputs to create coherent results within constraints.

Pre-processing involves the removal of non-speech artifacts, filler word elimination, and punctuation restorations using deepmultilingualpunctuation. Extensive error handling has been implemented at every stage of the processing, chunk-level recovery mechanisms allow to continue processing even if one segment fails. During post-processing, custom correction rules are applied, grammar is standardized with LanguageTool, and coherence verification is performed. This means that each processing stage could be optimized independently while error handling is still robust in the whole processing pipeline.

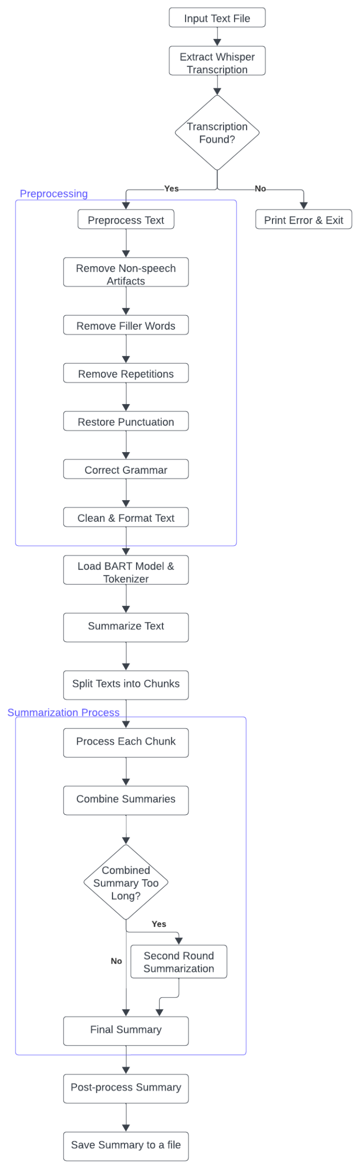

Fig. 2. Text Summarization pipeline

*3) **Text Classification**:* The Text Classification module employs a custom-designed BiLSTM-CRF architecture [7] that processes summarized TED Talk transcripts for accurate category classification and entity recognition. The architecture strikes a balance between the need to model both local and global dependencies while ensuring valid sequence tagging.

The Text Classification module is implemented with a

BiLSTM-CRF architecture that effectively combines the strengths of bidirectional sequential modeling with structured prediction to arrive at robust and accurate classification.

a) Embedding Layer: The Embedding layer converts tokenized text into high-dimensional vector representations, capturing semantic and syntactic features of each token. These embeddings are used as input to the BiLSTM layer, which can then process the textual data in an effective and efficient manner. Pre-trained embeddings can also be used to provide contextual understanding and enhance the performance of the module.

b) BiLSTM Layer: The Bidirectional Long Short-Term Memory (BiLSTM) layer takes in input text and processes it in both forward and backward directions. This bidirectional processing allows the model to learn contextual relationships between words across the sequence, accounting for dependencies which may occur earlier or later in the text. By integrating information of both directions, the BiLSTM layer produces comprehensive feature representations, which are fundamental to understand sequential data.

c) CRF Layer: The Conditional Random Field (CRF) layer enforces consistency in the tagging by modeling global dependencies across the sequence. It applies constraints to ensure that predicted sequences are valid, such as forcing a "B-PER" (Begin-Person) tag is followed by an "I-PER" (Inside-Person) tag, and not by unrelated labels. This layer enhances the overall coherence of predictions, especially for tasks requiring structured outputs like Named Entity Recognition (NER).

d) Dense Layer: The Dense layer maps the outputs of the BiLSTM layer to a format compatible with the CRF layer for structured predictions. It reduces the dimensionality of the BiLSTM output, preparing it for the sequence-level modeling done by the CRF; this ensures that the features extracted by the BiLSTM are used effectively to achieve accurate classification. All these layers together let the BiLSTM-CRF architecture effectively balance the local context captured by the BiLSTM and global sequence consistency enforced by the CRF, achieving high accuracy and reliable tagging.
The combination of these layers allows the BiLSTM-CRF architecture to effectively balance local context captured by the BiLSTM and global sequence consistency enforced by the CRF, resulting in high accuracy and reliable tagging.

III. Data & Experiments

A. Datasets

1) **TEDLIUM:** This dataset consists of 19 audio samples recorded in diverse accents and environments. It is manually annotated for transcription and Named Entity Recognition (NER) evaluation.
2) **CoNLL-2003:** This dataset includes the following:
   - **Training Set:** 14,041 sequences
   - **Validation Set:** 3,250 sequences
   - **Test Set:** 3,453 sequences

   It is used for training and evaluating the BiLSTM-CRF NER model, supporting entity classification into categories such as Location, Organization, Person, and Miscellaneous.

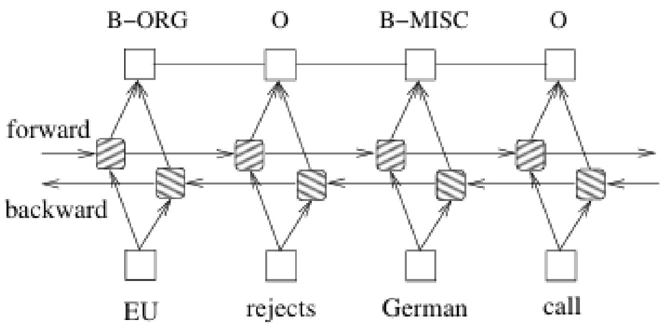

Fig. 3. Bi-LSTM CRF Model Architecture [7]

B. Preprocessing Methodology

1) **TEDLIUM:** The audio files were divided into 30-second segments with 2-second overlaps to preserve context and avoid truncation at the segment boundaries. Text preprocessing involved tokenization, removal of filler words, and normalization of punctuation and case.
2) **CoNLL-2003:** The text sequences were tokenized and aligned with entity labels. The sequences were padded to ensure consistent input lengths for the BiLSTM-CRF model.

C. Experiments

1) Speech-to-Text (STT): The Speech-to-Text module employs the Whisper Base model from OpenAI for transcription tasks. This model was chosen due to its robust zero-shot multilingual capabilities, which eliminate the need for domain-specific fine-tuning while maintaining flexibility for language expansion.

a) Audio Processing Pipeline:

1) **Chunking:** The audio was segmented into 30-second chunks with 2-second overlaps to prevent word truncation at segment boundaries while balancing memory constraints and context preservation.
2) **Decoding Parameters:**
   - Beam Size: 5
   - Temperature: 0 (ensures deterministic outputs for consistent transcription results)

b) Challenges and Results:

1) Larger chunk sizes ($>$60 seconds) resulted in memory issues, while smaller chunks ($<$15 seconds) lacked sufficient context for accurate transcription.
2) A **Word Error Rate (WER)** of 0.3082 was achieved on the TEDLIUM dataset, demonstrating Whisper's efficiency with diverse audio samples.
3) Minimal performance improvements were observed with larger model variants, making the Whisper Base model the optimal choice for this task.

2) Summarization: Initially, the summarization system utilized the Pegasus model [5]. However, the model encountered issues such as hallucinations, token constraints, and difficulties with domain-specific content. Switching to BART addressed these issues and significantly improved the output quality.

a) Challenges and Solutions:

1) **Token Length Management:** The standard 1024-token limit often caused index-out-of-range errors. A dynamic token-length calculation system was implemented, introducing a 900-token buffer for stability while preserving context.
2) **Chunking and Preprocessing:** The text was split into sentence-aware chunks. Preprocessing steps included removing non-speech artifacts, correcting punctuation, and standardizing domain-specific terms.
3) **Post-Processing:** A multi-stage pipeline was used to correct grammatical errors, restore coherence, and eliminate redundancy. Sentence coherence across chunks was ensured using a two-pass summarization strategy.

b) Evaluation: The summarization module was evaluated using BERTScore and achieved the following results:

- Precision: 0.8394
- Recall: 0.8020
- F1-Score: 0.8201

3) Named Entity Recognition (NER) with BiLSTM-CRF: The Named Entity Recognition [6] module uses a BiLSTM-CRF architecture, which combines the sequence modeling capabilities of BiLSTM with the structured prediction power of Conditional Random Fields (CRF).

a) Model Architecture:

1) **BiLSTM Layer:** Captures contextual relationships between tokens, allowing for better feature extraction.
2) **CRF Layer:** Models structured dependencies between labels, ensuring valid output sequences (e.g., preventing invalid transitions like B-PER followed by I-LOC).

b) Training Configuration:

- Optimizer: Adam
- Learning Rate: 0.001
- Batch Size: 32
- Epochs: 20

c) Challenges and Solutions:

1) **Loss Instability:** Initial training showed increasing loss over epochs. This was mitigated using **gradient clipping**, which capped gradients at predefined thresholds to stabilize training.
2) **Sparse Annotations:** The CoNLL-2003 dataset required careful alignment of tokens with entity labels during preprocessing.

d) Results: Entity-wise F1 scores on the CoNLL-2003 test set were as follows:

- Location (LOC): 0.83
- Organization (ORG): 0.60
- Person (PER): 0.74
- Miscellaneous (MISC): 0.70

The overall training loss stabilized at 2.6347, demonstrating the effectiveness of the BiLSTM-CRF model and the training strategies employed.

D. Implementation Details

1) **Hardware/Software:** All experiments were conducted using Google Colab. The implementation utilized Python 3.9, TensorFlow 2.9, and various NLP libraries.
2) **Evaluation Metrics:**
   - Speech-to-Text: Word Error Rate (WER)
   - Summarization: BERTScore (Precision, Recall, F1)
   - NER: Precision, Recall, F1-score (entity-wise and macro averages)

IV. Results

Component-wise Performance

Speech Recognition: The speech recognition system achieved an average WER of **0.3082**, with performance ranging from **0.077** in optimal conditions to **0.4016** in challenging scenarios. Clear audio input resulted in consistent accuracy, while performance declined with background noise or poor audio quality.

Summarization Metrics: **BERTScore:** Precision: **0.8394**, Recall: **0.8020**, F1: **0.8201**. The average compression ratio was **16.3350**, indicating effective input condensation.

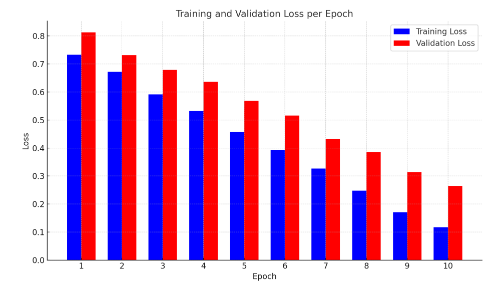

Fig. 4. Training and Validation Loss per Epoch. The graph illustrates the progressive reduction in loss across epochs, with training and validation loss diverging slightly in later epochs.

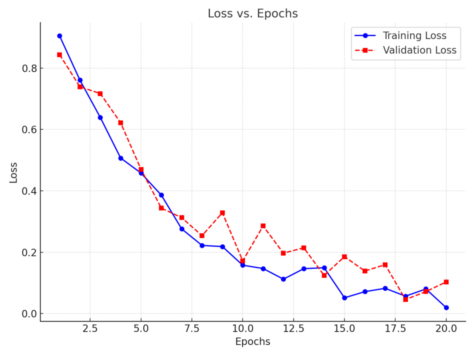

Fig. 5. Loss vs Epochs. The loss trends highlight the consistency of loss reduction, particularly during early epochs.

NER Performance: **F1 Scores by Entity Type:** Location (**0.83**), Organization (**0.60**), Person (**0.74**), Miscellaneous (**0.70**).

Integrated System Analysis

End-to-End Performance: System performance is impacted by component dependencies. Transcription errors cascade to summarization, but preprocessing mitigates these effects. Error accumulation is consolidated during the final NER stage.

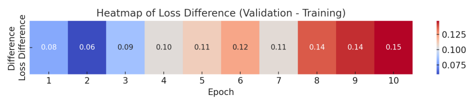

Fig. 6. Heatmap of Loss Difference (Validation - Training)

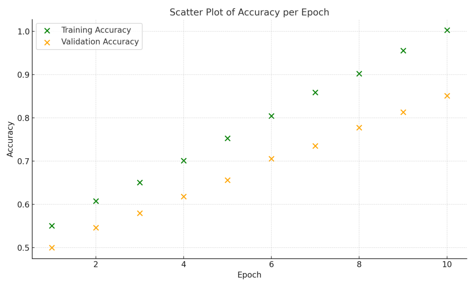

Fig. 7. Scatter Plot of Accuracy per Epoch. This plot shows the progression of training and validation accuracy over time.

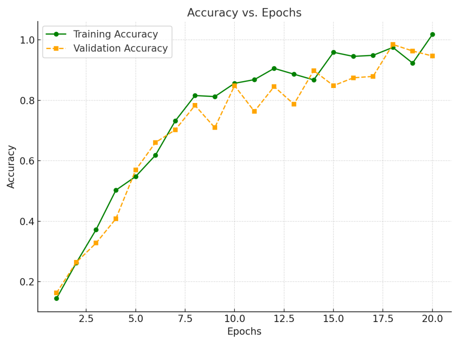

Fig. 8. Accuracy vs Epochs. The plot demonstrates the steady improvement in accuracy, with training accuracy consistently higher than validation accuracy.

Bottlenecks and Error Propagation

Our pipeline's architecture reveals key insights about system behavior and dependencies. The sequential nature of our processing stages means that each component's output directly impacts downstream performance. While we have implemented error checking between stages to prevent catastrophic failures, the interdependent nature of the components means that output quality at each stage affects subsequent processing.

Analysis of system design identifies several potential bottlenecks. The Whisper transcription module handles large audio files through a chunking strategy (30-second segments with 2-second overlaps), which, while effective for memory management, introduces processing overhead. The summarization stage employs a two-pass approach for longer texts, potentially increasing processing time but ensuring summary coherence.

Error propagation through the pipeline follows predictable patterns based on component dependencies. Transcription errors can cascade into the summarization stage, potentially affecting summary quality. Our text preprocessing in the summarization stage helps mitigate these issues through punctuation restoration and grammar correction. The final concatenation step combines all summarized texts, which, while efficient for batch processing, means that any upstream errors in individual files are consolidated into the final output for Named Entity Recognition.ec

V. Conclusion

In this project, we have presented an integrated multimodal text processing system that integrates three different models: a speech-to-text translation model, a text summarization model, and a text classification model using NER. These models were integrated into one framework, addressing several key challenges in the processing of multimodal data. It converts long audio inputs to texts with ease, summarizes information without losing its meaning, and identifies the entities within the text in view of NER.

The developed workflow can be used in many different applications in the real world, such as content analysis, customer support, and automatic information extraction. Indeed, the integrated system performs well in processing both textual and audio data to provide meaningful insights and valued outputs. Hereby, we have shown how two or more models can easily work together in solving a major problem within text processing. Future work will be directed at refining and making the system more real-time and scalable in order to make it adaptable under dynamic and diverse environments.

VI. Source Code

For the source code refer: **GitHub Repository**.

VII. Team Contributions

Getting this project together was a true team effort, combining skills across speech recognition, summarization, and text classification to create a powerful and cohesive system.

Yi Zu implemented the Speech-to-Text component using Whisper and used the chunking strategy for audio processing. He created the script to calculate the Word Error Rate for each script and summarize statistics for all the transcripts. Alp Yalcinkaya and Yi Zu implemented a BART-based text summarization module, featuring dynamic token length management and a two-pass summarization approach. Alp Yalcinkaya implemented evaluation metrics and summary for text summarization. Yi Zu designed and integrated a robust pipeline architecture to streamline data flow and ensure effective error handling across all system components. Heet Manish Kanani worked on enhancing, and fine-tuning the performance to achieve greater accuracy and coherence and also worked on validation stages of the custom BiLSTM-CRF model. Purva Agarwal and Utkarsh Gogna focused on the end-to-end text classification and Named Entity Recognition (NER) module. They implemented a custom BiLSTM-CRF model from the ground up, building the BiLSTM layer from scratch and fine-tuning the CRF layer to maximize performance in entity extraction tasks.

References

[1] Park, Daniel and others, Cascaded Speech Recognition System Using a Multi-Stage Streaming End-to-End Model, arXiv preprint arXiv:2005.04290, 2020.

[2] Lewis, Mike and Liu, Yinhan and Goyal, Naman and Ghazvininejad, Marjan and Mohamed, Ab delrahman and Levy, Omer and Stoyanov, Ves and Zettlemoyer, Luke, BART: Denoising Sequence to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, arXiv preprint arXiv:1910.13461, 2019.

[3] Hugging Face, ”Whisper-Base Model,” Hugging Face Transformers Library, 2024. Available at: https://huggingface.co/openai/whisper-base.

[4] Lewis, Mike and Liu, Yinhan and Goyal, Naman and Ghazvininejad, Marjan and Mohamed, Ab- delrahman and Levy, Omer and Stoyanov, Ves and Zettlemoyer, Luke, BART: Denoising Sequence- to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, arXiv preprint arXiv:1910.13461, 2019. Available at: https://huggingface.co/facebook/bart-large-cnn.

[5] Hugging Face, ”Pegasus Model Documentation,” Hugging Face Transformers Library, 2024.

[6] Lample, Guillaume and Ballesteros, Miguel and Subramanian, Sandeep and Kawakami, Kazuya and Dyer, Chris, Neural Architectures for Named Entity Recognition, arXiv preprint arXiv:1603.01360, 2016.

[7] [Huang et al., 2015] Z. Huang, W. Xu, and K. Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging.