TED UNIVERSITY

CMPE 492

Senior Project II

PROJECT NAME

Development of a Turkish Language Processing System with Integrated Speech Recognition and Synthesis

Final Report

Submission Date: 26.05.2025

Team Members:

Doğa Paksoy 10078725538

Rana Çetinkaya 31726748334

Berrin Uzun 15982088236

Table of Contents

1. Abstract

Development of a Turkish Language Processing System with Integrated Speech Recognition and Synthesis Project, aims to design, develop, and test a Turkish language system composed of Speech-to-Text (STT) and Text-to-Speech (TTS) components. The main objective is to ensure the accuracy, fluency, and robustness of these modules, especially under different acoustic conditions and dialectal variations. The test plan includes unit, integration, system, performance, user acceptance, and beta testing phases. Key areas evaluated include audio clarity, transcription precision, noise resilience, and real-time performance, especially in integration with FNSS services.

2. Introduction

In recent years, technologies for voice interaction have become increasingly significant across multiple industries. Nevertheless, there exists a considerable gap in the provision of high-performance systems specifically designed for the Turkish language. The majority of current speech processing tools either offer limited support for Turkish or fail to deliver the accuracy and naturalness necessary for a smooth user experience. This situation underscores the urgent requirement for a comprehensive solution capable of synthesizing natural-sounding Turkish speech and accurately converting spoken Turkish into written text.

This project seeks to address this issue by creating a thorough language processing system that effectively combines text-to-speech (TTS) and speech-to-text (STT) functionalities. The solution will undergo offline fine-tuning to enhance performance specifically for Turkish, and it will utilize sound isolation methods to guarantee clarity and minimize noise interference.

This project holds particular significance for FNSS services, where effective and dependable voice-based communication can greatly improve operational efficiency and user satisfaction. The project's scope includes model training, quality assessment, and integration into FNSS systems, emphasizing offline deployment and language-specific optimization, rather than focusing on real-time multilingual applications.

The primary goals of this project are:

- To create a natural-sounding Turkish text-to-speech (TTS) system.

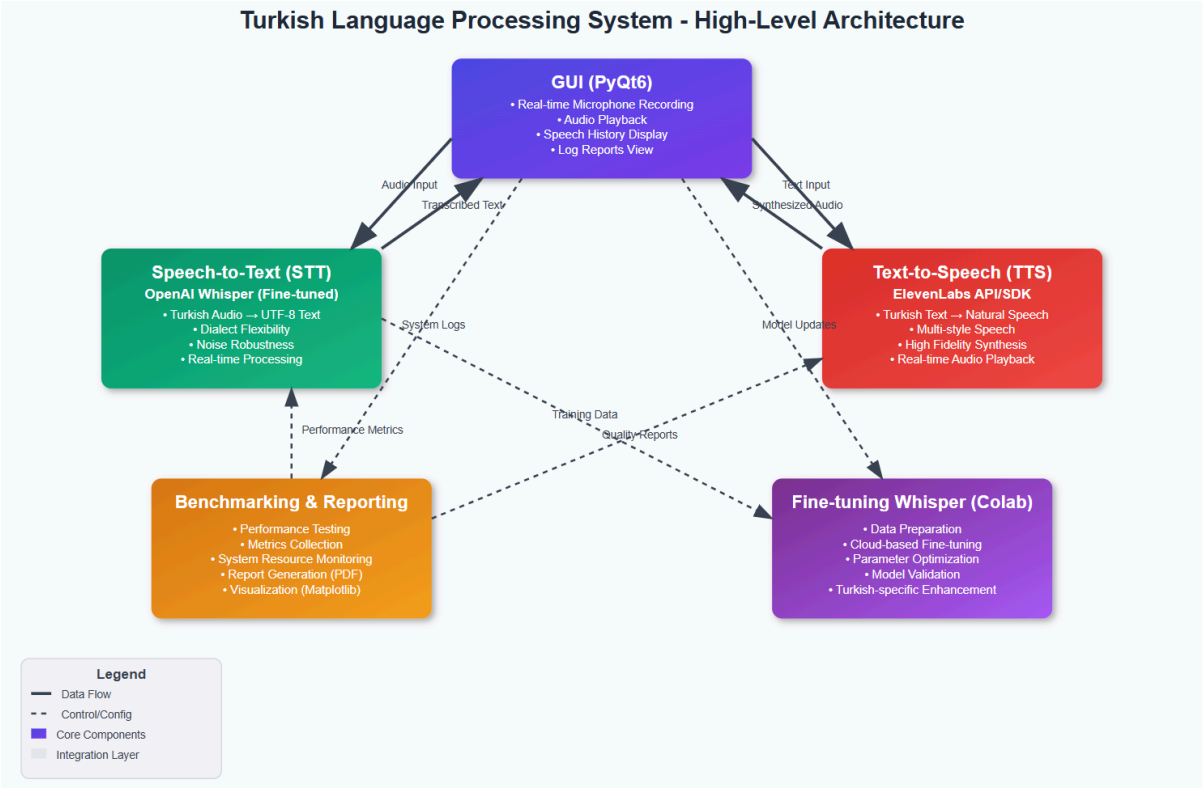
- To establish an accurate and dependable speech-to-text (STT) model for Turkish.
- To conduct offline fine-tuning of models specifically designed for Turkish linguistic characteristics.
- To evaluate performance through both quantitative and qualitative assessment metrics.
- To implement sound isolation methods to guarantee clear and understandable audio outputs.

3. System Architecture & Design

3.1 System Overview

The Turkish Language Processing System comprises five essential components to deliver a comprehensive solution for voice-text interactions in Turkish: a graphical user interface (GUI), speech-to-text (STT), text-to-speech (TTS), benchmarking and reporting, as well as model fine-tuning utilizing Whisper. This system is optimized for offline use, crafted for smooth FNSS integration, and guarantees precise, rapid, and noise-resistant performance.

3.2 High-Level Architecture Diagram



3.3 Component Descriptions

3.3.1 GUI (Graphical User Interface)

Function: Offers a simplified and user-friendly interface developed with PyQt6, enabling users to capture voice, hear system-generated replies, examine test logs, and retrieve speech history.

Key Features:

- Real-Time Microphone Recording

  Users have the ability to start live audio recording via their microphone. The system promptly processes the input through the STT module.

- Generated Audio Playback

  Once the TTS module generates a response, users have the ability to listen to the synthesized Turkish audio directly through the interface.

- Speech History Display

  The graphical user interface (GUI) keeps an ongoing record of previous interactions, which includes:

  - ➔ Timestamp
  - ➔ Transcribed speech (text output from speech-to-text technology)
  - ➔ Generated response (text input for text-to-speech technology)
  - ➔ Related audio playback controls
- View Log Reports

  Test and system logs are accessible via a distinct section in the GUI. These logs offer valuable insights into recognition accuracy, response times, and the overall status of the system.

Technology Stack:

- GUI Framework: PyQt6

- Audio Processing: Libraries based on Python, including sounddevice, wave, and pydub, facilitate real-time audio capture and playback.
- Module Integration: Direct function calls and local module bindings for STT, TTS, benchmarking, and fine-tuning components.
- Design Guidelines: A tidy layout, simplistic design, and a responsive interface tailored for desktop utilization.

3.3.2 Speech-to-Text (STT) Module

Function: The STT module is tasked with transforming spoken Turkish audio into a format that machines can read. It is engineered to accommodate real-time inputs while ensuring high accuracy across different dialects and varying noise conditions.

Core Model:
OpenAI Whisper (fine-tuned on Turkish data)

Output:

- Clean, UTF-8 encoded Turkish text.
- Appropriate for additional processing or immediate application in TTS and reporting modules.

Advantages:

- Dialect Flexibility:

  Capable of identifying and transcribing various Turkish accents and speaking styles.

- Noise Robustness:

  Operates consistently even in settings with slight background noise, owing to Whisper's pre-trained abilities.

- Integration:

  Effortlessly incorporated into the GUI for immediate transcription and logging.

3.3.3 Text-to-Speech (TTS) Module

Function: The TTS module transforms Turkish text into speech that sounds natural, facilitating voice-based feedback and interaction.

Core Technology:

ElevenLabs API/SDK for text-to-speech synthesis.

Key Features:

- Natural Voice Quality:

  ElevenLabs generates exceptionally realistic and expressive Turkish speech characterized by natural intonation and clarity.

- Multi-style Speech:

  It accommodates a range of speaking styles and emotions, allowing the output voice to be tailored to the specific application context.

- Real-time Audio Playback:

  The synthesized audio is streamed or played back directly within the GUI, eliminating the need for intermediate file downloads.

- Seamless Integration:

  The TTS module connects directly with the GUI through Python bindings, facilitating immediate response generation after transcription.

Advantages:

- High Fidelity Speech Synthesis:

  Enhances the user experience by providing lifelike voice outputs that enhance accessibility and engagement.

- Efficient Processing:

  Rapid synthesis designed for real-time interaction scenarios.

- Robust Turkish Language Support:

  Customized for Turkish phonetics, guaranteeing precise pronunciation of language-specific words and names.

3.3.4 Benchmarking & Reporting Module

Function: This module evaluates the system's performance by running tests on STT and TTS components, collecting detailed metrics, and generating comprehensive reports to monitor and improve the system.

Functions and Workflow:

- Performance Testing:

  Runs established test cases to evaluate accuracy, processing speed, and system stability.

- Data Collection:

  Records comprehensive metrics for each operation, which include:

  - ➔ Timestamp: Precise time of the event.
  - ➔ Log Level: Severity or type of the log (e.g., INFO, ERROR)
  - ➔ Operation: The specific action being recorded (e.g., transcription, synthesis)
  - ➔ Response Time: Time taken for processing.
  - ➔ Clarity Score: Quality metric for the generated speech or recognized audio clarity.
  - ➔ Accuracy Score: Correctness of transcription or match percentage.
  - ➔ Status: Outcome status (e.g., success, failure)
  - ➔ Description: Explanatory message detailing the log event.
- System Resource Usage:
  - ➔ CPU utilization percentage
  - ➔ Memory consumption
  - ➔ GPU usage (if applicable)

- Reporting:

  Produces comprehensive human-readable reports and visual summaries (charts, graphs) available through the GUI, encapsulating test results and resource usage.

Technologies Used:

- Python logging framework customized to capture extended attributes.
- System monitoring tools to fetch CPU, memory, and GPU usage stats.
- Visualization libraries such as Matplotlib.
- Report generation in PDF format for easy sharing.

Advantages:

- Offers detailed insights into system performance across various conditions.
- Aids in pinpointing bottlenecks associated with processing duration or hardware usage.
- Allows for monitoring of transcription and synthesis quality over time.
- Supports debugging and focused enhancements based on recorded data.

3.3.5 Fine-tuning Whisper Module

Function: This module improves the performance of the foundational OpenAI Whisper model by conducting offline fine-tuning tailored specifically for the Turkish language and its domain-specific speech patterns.

Process:

- Data Preparation:

  Compiled a Turkish speech dataset complete with transcriptions, encompassing diverse dialects and acoustic environments, to accurately represent real-world usage.

- Offline Fine-tuning:

  Utilized transfer learning methodologies to modify Whisper's pre-trained weights for Turkish-specific phonetics and vocabulary. This process enhances transcription accuracy beyond the capabilities of the base model.

- Parameter Optimization:

  Modified hyperparameters such as learning rate, batch size, and number of epochs to achieve a balance between training duration and model performance.

- Validation:

  Evaluated fine-tuned models on distinct validation sets to avoid overfitting and to ensure effective generalization.

Advantages:

- Improved Accuracy:

  Customizing the model to accommodate the subtleties of the Turkish language greatly minimizes inaccuracies, especially concerning intricate phonemes and regional dialects.

- Robustness:

  Fine-tuning enhances Whisper's ability to handle domain-specific terms, background noise, and conversational speech.

- Efficient Deployment:

  Offline fine-tuning allows for the incorporation of model updates without the need for constant internet access, thereby promoting privacy and security.

3.4 Design Principles

The architecture of the system and the implementations of its components are directed by a number of essential design principles to guarantee reliability, scalability, and ease of use.

1. Modularity

Each primary functionality — GUI, STT, TTS, benchmarking — is developed as a separate module with clearly defined interfaces. This modular approach enables simpler maintenance, testing, and future enhancements.

2. Scalability

The design accommodates the potential expansion of the system to manage larger datasets, an increased number of users, and the integration of additional services (such as FNSS) without necessitating significant restructuring.

3. User-Centered Design

The graphical user interface (GUI) is crafted with an emphasis on simplicity and intuitiveness, catering to users from various technical backgrounds. Immediate feedback and well-defined interaction pathways enhance usability.

4. Robustness

The system utilizes the inherent noise resilience of Whisper along with the superior synthesis capabilities of ElevenLabs to ensure optimal performance in realistic, noisy settings. Logging and benchmarking modules facilitate monitoring, enabling the swift identification and resolution of issues.

5. Maintainability

The clear separation of code, thorough logging, and extensive testing contribute to effective debugging and gradual improvements.

4. Engineering Impact Analysis

1. Global Impact

The initiative advances speech-based interface technologies for underrepresented languages, such as Turkish, globally. The analysis and adaptation of speech and LLM models for Turkish helps reduce linguistic bias in AI systems.

2. Economic Impact

Through offline processing support and integration with FNSS services, the system can increase operational reliability in contexts with limited resources. In settings with sporadic connectivity, this could lead to greater efficiency.

3. Environmental Impact

The use of offline-capable models with optimized computation (e.g., inference on local machines with reduced internet dependency) minimizes energy use in cloud computation, promoting eco-efficient AI deployment in remote or bandwidth-limited environments.

4. Societal Impact

The system can be made more inclusive and accessible for a variety of Turkish-speaking people by supporting various Turkish accents and dialects and placing a strong emphasis on natural prosody and clarity.

5. Contemporary Issues

Privacy Issues in Voice Systems

Voice systems frequently handle sensitive user information, which raises concerns regarding data security and unauthorized access. Ensuring that the Turkish TTS and STT models function offline aids in alleviating these risks by maintaining data locally and privately.

Ethics of AI-Generated Speech

AI-generated speech should be easily identifiable as distinct from human voices to prevent misinformation or manipulation. The system must incorporate safeguards to avert misuse, particularly in sensitive defense-related contexts.

Real-World Applications

This initiative serves a functional purpose in FNSS services, facilitating clearer communication, enhanced accessibility, and increased operational efficiency via natural Turkish voice interaction.

Trends in NLP for Turkish

Recent developments in natural language processing have enhanced support for morphologically complex languages such as Turkish. Tailoring models specifically for Turkish leads to improved accuracy and relevance, in line with the ongoing trends towards AI solutions that are language-specific.

User Acceptance and Trust

Users might be reluctant to completely trust speech or transcription generated by AI, thus making transparency and reliability crucial.

6. Tools, Technologies, and Resources

This section outlines the key tools, technologies, and resources employed throughout the development of the Turkish Language Processing System.

6.1 Programming Languages and Frameworks

- Python:
  The main programming language utilized for backend development, model fine-tuning, and benchmarking scripts.
- PyQt6:
  Employed for creating the graphical user interface, offering a cross-platform framework for desktop applications.

6.2 Speech Processing Models and APIs

- OpenAI Whisper:
  State-of-the-art speech-to-text model refined offline to improve recognition of the Turkish language.
- ElevenLabs TTS:
  Utilized for text-to-speech synthesis, delivering realistic and expressive voice outputs in Turkish.

6.3 Libraries and Tools

- PyTorch:
  Employed for the training and fine-tuning of the Whisper model.
- Matplotlib:
  Applied for the creation of visual reports and performance graphs within the benchmarking module.
- Logging Module (Python standard library):
  Tailored for enhanced logging of system performance and operational metrics.

- System Monitoring Tools:
  Utilized for monitoring CPU, memory, and GPU usage throughout testing.

6.4 Resources for Background Research

- Scientific Articles and Papers:
  Consulted studies regarding speech recognition, text-to-speech technologies, and the phonetics of the Turkish language.
- Online Documentation and Tutorials:
  Utilized the official documentation for OpenAI Whisper, ElevenLabs API, and PyQt6.
- Open-source Codebases:
  Analyzed current projects and implementations to guide design decisions.

6.5 Infrastructure

- Local Development Environment:
  The primary system, which encompasses the GUI and benchmarking tools, was created and evaluated on local computers operating on Windows and Linux platforms.
- Google Colab (Fine-tuning Whisper):
  The fine-tuning of the Whisper model was conducted in the GPU-accelerated environment of Google Colab. This setup offered the essential computational power without the need for dedicated local hardware.
- Offline Deployment Capability:
  After fine-tuning, both the models and the application can be installed locally without the need for internet connectivity, thereby ensuring user privacy and maintaining strong performance in limited environments.

7. Testing and Results

Testing covered all major components including TTS and STT modules, noise isolation pipelines, FNSS API integration, and performance monitoring. The strategy involved unit testing, integration testing, system testing, performance evaluation, user acceptance testing, and beta deployment within FNSS environments.

Test Case Table:

Feature	Description	Expected Result	Actual Result	Status
TTS Model	Generate natural Turkish audio	Clear, accurate, accent-aware output	Met expectations	Pass
STT Module	Transcribe speech accurately	High transcription accuracy	Accurate for various dialects	Pass
Noise Isolation	Remove background noise	Maintain clarity in noisy environments	Effective in most cases	Pass
FNSS API Integration	Seamless data exchange	Low-latency, error-handled responses	Functional with minor issues	Pass
Offline Fine-Tuning	Adapt to domain-specific speech	Improved results post-tuning	Enhanced performance observed	Pass

Bug Discussion and Enhancements:

During the testing process, several issues were identified:

- **Audio Quality Under Noise:** In environments with high background noise, the speech-to-text (STT) accuracy declined, affecting transcription precision.
- **Dialectal Recognition:** The STT module encountered difficulty in accurately recognizing speech with less common Turkish regional dialects.
- **FNSS Integration Issues:** During API communication with FNSS platforms, occasional errors and latency problems were observed.

Enhancements implemented to address these issues include:

- Expanding training datasets with diverse Turkish dialects to improve recognition accuracy.

- Applying advanced noise isolation and filtering techniques to enhance STT reliability in noisy settings.
- Strengthening the FNSS API interface with better error handling and fallback mechanisms for unstable connections.

These mitigations aimed to improve system resilience, audio clarity, and robust integration performance.

8. Conclusion

This project has successfully created an integrated Turkish language processing system that combines speech-to-text and text-to-speech functionalities, specifically optimized for offline use. A high degree of accuracy and naturalness, specifically suited for the Turkish language, is achieved by the system through meticulous fine-tuning, benchmarking, and noise isolation procedures. The integration with FNSS services illustrates its practical applicability in real-world situations, thereby enhancing communication and operational efficiency. Despite the ongoing challenges posed by dialectal differences and noisy environments, continuous improvements are anticipated to enhance performance further. In conclusion, this research provides a significant, user-focused solution for Turkish speech technology, setting the stage for future progress in language-specific AI applications.

9. Appendices

Datasets Used:

- ysdede/commonvoice_17_tr_fixed:
  Cleaned and processed version of the Mozilla Common Voice project based on Turkish version 17. Used for Turkish speech recognition models.
- ysdede/khanacademy-turkish:
  It is a speech recognition dataset containing audio and transcripts of Khan Academy Turkish videos. It was used to increase model accuracy in training speeches.
- deniskaanalpay/tts-test2:
  It is a special dataset created for Turkish text-to-speech (TTS) tests, containing synthetic voice and written text pairings.

10. References

1. ysdede. *commonvoice_17_tr_fixed*. Hugging Face. https://huggingface.co/datasets/ysdede/commonvoice_17_tr_fixed

2. ysdede. *khanacademy-turkish*. Hugging Face. https://huggingface.co/datasets/ysdede/khanacademy-turkish

3. deniskaanalpay. *tts-test2*. Hugging Face. https://huggingface.co/datasets/deniskaanalpay/tts-test2