TED UNIVERSITY

CMPE 491

Senior Project I

PROJECT NAME

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

Project Specifications Report

Submission Date: 25.10.2024

Team Members:

Doğa Paksoy 10078725538

Rana Çetinkaya 31726748334

Berrin Uzun 15982088236

Table of Contents

1 Introduction

1.1 Description

This project aims to develop sophisticated AI model called the Turkish Language Processing System. This system functions as a dual-purpose AI model and is intended for both text-to-speech (TTS) and speech-to-text (STT) functions in the Turkish language. The system accurately converts spoken language into text and converts written text into natural-sounding voice using machine learning techniques and natural language processing (NLP). This AI model integrates smoothly with FNSS services by offering an effective interface for voice-based operations. Aims to improve user experience and streamlining workflows. In order to provide a superior, culturally sensitive user experience, the AI model includes many aspects of fine-tuning for the unique characteristics of Turkish.

1.2 Constraints

- Language-Specific Model Tuning: Due to Turkish's unique phonetic and syntactic structures, extensive customization and fine-tuning are required.
- Data Quality and Quantity: Robust STT and TTS functionalities require large, high-quality datasets that represent various Turkish dialects, accents, and vocabulary.
- Noise Reduction: The integration of sound isolation technology to filter out background noise while maintaining speech quality is crucial but it is technically challenging to integrate. Effective implementation will require fine-tuning algorithms for various acoustic environments.
- Processing and Latency Constraints: The system must process and respond in real time to ensure a flawless user experience. Operational effectiveness will depend on maintaining a balance between processing speed and model complexity, particularly in resource-constrained environments.
- Offline Functionality: The deployment strategy of the model becomes more complex when offline capabilities are implemented for areas with limited internet connectivity, necessitating small, lightweight models without compromising accuracy.

1.3 Professional and Ethical Issues

Developing a Turkish Language Processing System with Integrated Speech Recognition and Synthesis raises several professional and ethical considerations, particularly regarding user privacy, data security, and the accurate representation of Turkish language nuances.

Privacy and Security of Data:

Text-to-speech and speech-to-text systems frequently deal with private user information, including personally identifiable information. It is crucial to protect the integrity and confidentiality of audio data. Strong data encryption procedures will be put in place to address this, and robust data management guidelines will be adhered to safeguard user privacy. In keeping with FNSS's dedication to data security and user trust, offline processing capabilities are given priority to reduce the risks of data exposure.

Fairness and Accessibility:

To avoid biases in voice recognition, it is crucial to make sure that the system's language processing algorithms function correctly across a range of Turkish dialects and accents. To prevent the inadvertent exclusion of regional dialects, ethical considerations must also be considered when choosing training data. Furthermore, this system seeks to improve digital interaction for people with speech or literacy impairments, thereby promoting inclusivity in technology.

Transparency and Consent:

Before their voice data is processed, users should be made aware of the data collection procedures and give their consent. By ethical AI principles, a transparent and easy-to-use consent mechanism will be incorporated to guarantee data use transparency.

Professional Accountability:

In an effort to create a dependable and accountable system, the project team is dedicated to following the ethical standards and industry norms set forth by IEEE. To guarantee the integrity of the system and adherence to professional codes of conduct, proper documentation, ongoing model evaluation, and development accountability are given top priority.

2 Requirements

Hardware Requirements

Processing Unit:

To manage machine learning tasks and maximize processing speed, a strong central and graphics processor is required.

Memory:

A sufficient memory space to manage sizable datasets and facilitate smooth model operations.

Storage:

Additional storage for model files and datasets, as well as fast primary storage for instant data access.

Audio Equipment:

Accurate audio input and output during testing are ensured by dependable noise-canceling microphones and headsets.

Additional Hardware:

The ability to access scalable cloud computing resources to facilitate extensive training and optimization, if necessary.

Software Requirements

Text-to-Speech (TTS) Engine:

An advanced text-to-speech engine that can generate natural-sounding speech from Turkish text. A variety of voices and accents should be supported by the engine to improve user experience.

Speech-to-Text (STT) Engine:

A powerful STT engine that generates written text from spoken Turkish with accuracy. For the engine to provide transcription accuracy, it must be able to handle a variety of accents and dialects.

Machine Learning Framework:

Optimizing performance for the Turkish language by developing, training, and fine-tuning the TTS and STT models using a machine learning framework.

Noise Isolation Algorithms:

Use sound isolation techniques to reduce background noise and improve the clarity of output speech for TTS and input speech for STT.

Data Preprocessing Tools:

Tools to ensure high-quality input for TTS and STT processes by preprocessing and cleaning text and audio data.

Offline Fine-Tuning Capabilities:

The models' performance and adaptability to different contexts can be enhanced by offline fine-tuning with local datasets specific to the Turkish language.

Performance Benchmarking Tools:

The efficiency of the TTS and STT systems is assessed using performance and quality benchmarking tools, which produce comprehensive results on criteria including accuracy, speed, and user satisfaction.

Integration APIs:

APIs to enable efficient communication and data sharing between the TTS/STT system and other operational tools, as well as seamless connection with FNSS services.

User-Friendly Interface:

Users can easily engage with the TTS and STT features using an intuitive web-based interface that makes input, output playback, and system monitoring simple.

Alert & Notification System:

A system that will notify users in real time of transcription or synthesis errors, making sure they are aware of problems as soon as they arise.

Scalability:

To support diverse user requirements and operational situations, the system must be built to scale effectively and manage fluctuating amounts of text and speech data.

Data Security Measures:

Protect user data and preserve confidentiality during processing and storage by putting security measures in place, such as access limits and encryption.

Algorithm Requirements

The project will use cutting-edge machine learning methods to enable speech-to-text (STT) and text-to-speech (TTS) capabilities. These algorithms are crucial for correctly translating spoken language into text and written text into natural-sounding speech.

Deep Learning Algorithms:

Deep learning algorithms will be used to improve the TTS and STT systems' performance, especially those made for audio and natural language processing (NLP). For STT applications, recurrent neural networks (RNNs), particularly Long Short-Term Memory (LSTM) networks, are perfect since they excel at sequence prediction tasks.

Convolutional Neural Networks (CNNs):

CNNs are useful for identifying patterns in high-dimensional data, therefore they can also be used for audio feature extraction. CNNs can help process audio input spectrograms in this project to increase the accuracy of speech recognition.

Implementation Libraries:

The deep learning libraries required for the system will be implemented using the following libraries:

- ➔ TensorFlow: The open-source TensorFlow deep learning framework was created by Google and is known for its adaptability and scalability. It will be used for both TTS and STT model building and training. TensorFlow can facilitate STT implementation by offering pre-trained models and the TensorFlow Speech Recognition API.
- ➔ Keras: Keras is a high-level API that runs on top of TensorFlow and is used for building deep learning models. Its wide library of pre-built layers and functions, along with its user-friendly interface, will provide quick prototyping and development of TTS and STT systems.
- ➔ PyTorch: Facebook's PyTorch is very helpful for research and development since it provides dynamic computing graphs. This library will be used to improve the performance of TTS and STT systems and to experiment with different model designs.

Data Preprocessing Algorithms:

Strong data preprocessing algorithms will be used to prepare and clean audio and text data. The TTS and STT models will use methods like noise reduction, normalization, and tokenization to improve the quality of their inputs.

Benchmarking Algorithms:

The TTS and STT systems' accuracy, speed, and user satisfaction will be assessed using performance and quality benchmarking techniques. To evaluate system performance, metrics like the Naturalness Score for TTS and the Word Error Rate (WER) for STT will be computed.

3 References

[1] Code of Ethics