TED UNIVERSITY

CMPE 492

Senior Project II

PROJECT NAME

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

Low-Level Design Report

Submission Date: 09.03.2025

Team Members:

Doğa Paksoy 10078725538

Rana Çetinkaya 31726748334

Berrin Uzun 15982088236

Table of Contents

1 Introduction ……...3

1.1 Object design trade-offs ……3

1.2 Interface documentation guidelines ...……...3

1.3 Engineering standards ……....4

1.4 Definitions, acronyms, and abbreviations ……..……..6

2 Packages ……...7

3 Class Interfaces ……...9

3.1 Speech-to-Text (STT) ……..……..……9

3.2 Text-to-Speech (TTS) ……..……...10

3.3 Noise Reduction & Sound Isolation ……..……11

3.4 Data Integration (Dialect Adaptation) ……..……...12

3.5 Reporting & Benchmarking ……...……..……13

3.6 API Integration & FNSS Compatibility ……..……..13

4 Glossary ……..……14

5 References ……...16

1 Introduction

1.1 Object design trade-offs

A number of trade-offs need to be taken into account when creating the object-oriented framework for the Turkish Language Processing System with Integrated Speech Recognition and Synthesis:

- **Performance vs. Maintainability:** While highly optimized data structures speed up systems, they can also make them more complex, which makes debugging and maintenance more difficult.
- **Efficiency vs. Flexibility:** While a modular architecture permits future extensions (like the addition of other languages), it may result in an increase in execution time.
- **Processing Speed vs. Memory Usage:** While keeping big precomputed models in memory improves response time, it also uses more resources.
- **Scalability vs. Simplicity:** While a highly scalable system may accommodate higher utilization, it necessitates a more complex architectural design.
- **Security vs. Usability:** Access controls and encryption safeguard private information, but they also make user authentication and data retrieval more difficult.

The system maintains a structured object-oriented design while guaranteeing robust performance, scalability, and user-friendliness by striking a balance between these aspects.

1.2 Interface documentation guidelines

The Turkish Language Processing System uses an API-based methodology to guarantee smooth data interchange with FNSS services.

- **Data Formats:** JSON will be used by the system to exchange structured data.
- **Authentication:** Token-based authentication techniques will be used to safeguard API access, guaranteeing that only systems and users with permission can communicate with the services. To safeguard user information and uphold system integrity, secure credential management will be used.
- **Error Handling:** Issues will be reported using standard HTTP status codes and thorough error messages.

- **Input and Output Data:** The anticipated input and output of each component will be precisely recorded.

1.3 Engineering standards

This project complies with recognized software engineering standards to guarantee maintainability, scalability, and adherence to security and privacy regulations. Below is a comprehensive overview of the standards implemented and their significance to the project.

UML (Unified Modeling Language)

UML serves as a standardized modeling language for visualizing, specifying, and documenting the architectures of software systems. In this project, UML diagrams are utilized for:

- Class Diagrams: To depict the relationships among system components.
- Sequence Diagrams: To demonstrate interactions and data flow between various modules.
- Component Diagrams: To delineate dependencies and the functions of different subsystems.
- Data Flow Diagrams (DFD): To illustrate the movement of data throughout the system.

The application of UML guarantees that the software architecture is thoroughly documented, facilitating a clearer understanding of the system for both developers and stakeholders.

IEEE 830-1998 - Software Requirements Specification (SRS) Standard

The IEEE 830-1998 standard offers a framework for documenting software requirements. In this project, the system's requirements are articulated in alignment with this standard. The requirements documentation encompasses:

- Functional Requirements: Outlining the essential functionalities of the system, including STT and TTS accuracy and speed.
- Performance Requirements: Establishing limits on response time, computational capacity, scalability, and resource utilization.

- Security Requirements: Specifying data protection strategies, user authentication protocols, and compliance with GDPR/KVKK.

By adhering to this standard, we ensure that the system's requirements are explicitly defined and systematically organized.

IEEE 1016-2009 - Software Design Document (SDD) Standard

The IEEE 1016-2009 standard delineates the structure for documenting software design. This low-level design (LLD) report adheres to this standard by encompassing:

- System Architecture: An analysis of the overall framework of the software and its constituent components.
- Database Model: Information regarding the storage and management of persistent data.
- Interface Definitions: Explanations of the interactions and communications between modules and external services.
- Control Flow: Logical sequences that illustrate the system's operation across various scenarios.

This standard is instrumental in fostering a coherent and scalable software design.

GDPR & KVKK Compliance - Data Security and Privacy Protection

The project is structured to align with the General Data Protection Regulation (GDPR) and Turkey's Personal Data Protection Law (KVKK), thereby safeguarding user data security and privacy.

- Data Encryption: All sensitive user information is encrypted during both storage and transmission.
- Authentication & Authorization: Mechanisms for access control are established to ensure that users can only access data for which they are authorized.
- Data Retention Policies: User data is retained solely for the necessary duration to mitigate privacy risks.
- Anomaly Detection & Security Monitoring: Systems for continuous monitoring are implemented to identify and thwart security threats.

By adhering to these security protocols, the system guarantees the protection of user privacy and compliance with regulatory requirements.

1.4 Definitions, acronyms, and abbreviations

- **STT (Speech-to-Text):** A technology that transforms spoken words into written text.
- **TTS (Text-to-Speech):** A technology that converts written text into artificial speech.
- **NLP (Natural Language Processing):** A field of artificial intelligence dedicated to enabling computers to comprehend and produce human language.
- **FNSS:** FNSS Savunma Sistemleri A.Ş., an external partner specializing in speech processing applications.
- **API (Application Programming Interface):** A collection of functions and protocols that facilitate communication between different software applications.
- **UML (Unified Modeling Language):** A standardized language utilized for visualizing software design.
- **IEEE 830-1998:** A standard that specifies the format for software requirements documentation.
- **IEEE 1016-2009:** A standard pertaining to software design documentation.
- **GDPR (General Data Protection Regulation):** A regulation from the European Union that governs the collection, processing, and storage of personal data.
- **KVKK (Kişisel Verileri Koruma Kanunu):** Turkey's law on personal data protection, ensuring the privacy and security of data.
- **JSON (JavaScript Object Notation):** A lightweight format for data interchange, used for structuring and transmitting information.
- **Deep Learning:** A branch of machine learning that employs artificial neural networks to identify complex patterns within data.
- **Transformer Model:** An architecture in deep learning designed for applications such as speech recognition and natural language processing.
- **Benchmarking:** The practice of assessing system performance by comparing it against established metrics.
- **RESTful API:** An architectural approach for creating networked applications that utilize HTTP requests for communication.
- **Acoustic Model:** A deep learning model that correlates audio features with phonetic representations.

- **Vocoder:** A component in speech synthesis that translates acoustic parameters into actual audio waveforms.
- **Signal Processing:** A collection of techniques aimed at enhancing, analyzing, or manipulating audio signals.
- **Noise Filtering:** The technique of eliminating unwanted background noise to enhance audio clarity.

2 Packages

1. transformers

- Description: A library developed by Hugging Face designed for utilizing pre-trained large language models (LLMs) such as Whisper, BERT, GPT, and T5 for various natural language processing (NLP) tasks.
- Functions:
    - Facilitates the loading of transformer models for both speech-to-text and text-to-speech applications.
    - Enables the fine-tuning of LLMs specifically for Turkish speech recognition and synthesis.
    - Supports tasks such as text generation, summarization, and translation.
- Usage: This library is employed within the Speech-to-Text (STT) and Text-to-Speech (TTS) modules to adapt pre-trained models for the Turkish dialect.

2. torch (PyTorch)

- Description: A popular deep learning framework utilized for the training and deployment of neural networks.
- Functions:
    - Facilitates the training of deep learning models for both speech recognition and synthesis.
    - Offers GPU acceleration to enable real-time processing capabilities.
    - Allows for the fine-tuning of models based on transformer architecture.
- Usage: Serves as the main deep learning framework for all tasks related to speech processing.

3. torchaudio

- Description: A library built on PyTorch designed for the processing and analysis of audio signals.
- Functions:
    - Efficiently loads and processes speech waveforms.
    - Implements data augmentation methods, such as noise reduction and echo suppression.
    - Seamlessly integrates with deep learning models for speech recognition.
- Usage: This library is utilized in the Noise Reduction and Sound Isolation module, as well as for preprocessing speech input prior to its input into the Speech-to-Text (STT) model.

9. json

- Description: A native Python library designed for the management of JSON data.
- Functions:
    - Transforms Python objects into JSON format to facilitate API interactions.
    - Interprets JSON requests and responses.
    - Retains structured data pertaining to speech recognition outcomes.
- Usage: Employed for data interchange between system components and FNSS services.

10. pandas

- Description: A Python library designed for data manipulation and analysis.
- Functions:
    - Evaluates performance metrics for Speech-to-Text (STT) and Text-to-Speech (TTS) systems.
    - Assesses errors in speech recognition and compiles benchmarking outcomes.
    - Manages extensive datasets of Turkish speech corpora.
- Usage: This library is utilized within the Reporting & Benchmarking module to assess system performance.

3 Class Interfaces

3.1 Speech-to-Text (STT)

Accurately translating spoken Turkish into written text is the responsibility of the Speech-to-Text (STT) component. This module uses cutting-edge deep learning algorithms to guarantee excellent transcription accuracy, dialect adaptation, and real-time processing. This component consists of:

- **Audio processing:** Background noise is eliminated and clarity is increased by filtering incoming speech. Adaptive noise reduction methods improve speech quality and increase the accuracy of recognition.
- **Feature Extraction:** Using deep learning models trained on several Turkish voice datasets, the system extracts linguistic and phonetic features from the audio source.
- **Speech Recognition Model**: In order to handle differences in accents and dialects, a neural network-based approach translates retrieved characteristics to Turkish words and phrases.
- **Text Refinement:** Common speech-to-text errors are fixed, appropriate punctuation is applied, and fluency is ensured by processing the raw transcribed text.

With its high accuracy across multiple Turkish dialects and real-time transcription optimization, this component is robust for a wide range of use scenarios.

Attributes:

- language_model:SpeechRecognitionModel → The voice recognition deep learning model.
- noise_filter: NoiseReductionModule → Background noise filtering module.
- transcription_confidence: float → STT prediction confidence score.
- punctuation_module:TextRefinementModule → Manages text correction and punctuation.

Methods:

- process_audio(audio_input: AudioStream) -> TextOutput: This function transforms audio input into written text.

- filter_noise(audio_input: AudioStream) -> AudioStream: This function eliminates background noise prior to further processing.
- extract_features(audio_input: AudioStream) -> FeatureVector: This function retrieves linguistic and phonetic characteristics from the spoken input.
- generate_transcription(feature_vector: FeatureVector) -> str: This function translates speech features into textual format.
- apply_text_refinement(text: str) -> str: This function rectifies inaccuracies and ensures appropriate punctuation is applied.

3.2 Text-to-Speech (TTS)

The Text-to-Speech (TTS) feature ensures proper pronunciation, tone, and rhythm by translating Turkish text into speech that sounds natural. This component consists of:

- **Preprocessing Module:** Applies linguistic principles, eliminates inconsistencies, and tokenizes input text.
- **Acoustic Model:** Generates speech parameters using deep learning models.
- **Vocoder:** Creates real voice waveforms using generated parameters.

This part incorporates user-controlled adjustments and uses deep learning models to guarantee high-quality speech synthesis.

Attributes:

- voice_model: SpeechSynthesisModel → The advanced deep learning framework employed for generating speech.
- text_preprocessor: TextNormalizationModule → Processes and standardizes text input for clarity.

Methods:

- synthesize_speech(text_input: str) -> AudioStream → Transforms written text into audible speech.
- normalize_text(text_input: str) -> str → Processes input text by managing abbreviations and numerical values.

- convert_to_phonemes(text: str) -> PhonemeSequence → Translates text into phonetic forms.
- generate_audio(phonemes: PhonemeSequence) -> AudioWaveform → Transforms phonemes into sound waveforms.

3.3 Noise Reduction & Sound Isolation

The main goal of the Noise Reduction & Sound Isolation module is to improve the clarity of speech and the accuracy of recognition. This module includes the following components:

- **Noise Filtering:** Detects and eliminates background noise while maintaining the integrity of the speech.
- **Adaptive Filtering:** Adjusts in real-time to various acoustic settings to enhance the quality of speech input.
- **Echo Cancellation:** Minimizes reverberation and echoes in recordings, thereby enhancing speech accuracy.

Attributes:

- noise_filter_strength: float → Modifies the intensity of noise reduction.
- adaptive_filter_enabled: bool → Activates or deactivates real-time adaptive filtering.
- echo_cancellation_enabled: bool → Engages echo suppression functionality.

Methods:

- filter_background_noise(audio_input: AudioStream) -> AudioStream → Eliminates background noise interference.
- apply_adaptive_filtering(audio_input: AudioStream) -> AudioStream → Adjusts noise filtering dynamically according to input conditions.
- reduce_echo(audio_input: AudioStream) -> AudioStream → Mitigates reverberation and echo effects.

This module greatly enhances the reliability of both Speech-to-Text (STT) and Text-to-Speech (TTS) functionalities by providing clean and high-quality audio input.

3.4 Data Integration (Dialect Adaptation)

The system's capacity to accommodate various Turkish dialects is improved by the Data Integration Component. It improves voice recognition and synthesis for greater accuracy and inclusivity by processing several datasets, ensuring the system can manage regional accents and linguistic differences. Important processes include:

- **Dialect Data Collection:** Numerous voice datasets covering various Turkish accents are gathered, including crowdsourced data, publicly accessible Turkish speech datasets, and existing linguistic studies.
- **Model Training:** In an LLM setting, unsupervised learning is used to train transformer-based models on datasets specific to a certain dialect. This makes it possible for the model to identify regional linguistic patterns. The accuracy of recognition and synthesis is then increased by fine-tuning it to accommodate region-specific pronunciation variations.

This integration increases the system's speech recognition and synthesis capabilities and improves accuracy for users with various accents.The Data Integration element component ensures that the system will continue to be flexible, precise, and inclusive while offering efficient voice synthesis and recognition in all Turkish dialects.

Attributes:

- dialect_database: Dataset → Preserves linguistic diversity and regional accents.
- model_trainer: ModelTrainingModule → Optimizes models for various dialects.

Methods:

- collect_dialect_data(source: DataSource) -> Dataset → Gathers and stores regional speech data.
- train_model_on_dialect(dialect_data: Dataset) -> SpeechRecognitionModel → Adapts STT/TTS models to dialects.
- apply_dialect_correction(text_input: str, dialect: str) -> str → Refines speech recognition based on regional linguistic patterns.

3.5 Reporting & Benchmarking

Tools for measuring and evaluating the effectiveness of the TTS and STT functions are available in the Reporting & Benchmarking component. Important characteristics include:

- **Performance Metrics:** Assesses the system's clarity, latency, and accuracy.
- **Automated Benchmarking:** Produces comparative reports by running pre-defined test cases.
- **Report Generation:** Enables users to export and filter reports according to metrics and time periods.

The system maintains high dependability by combining benchmarking and reporting, guaranteeing ongoing language processing model improvement.

Attributes:

- performance_metrics: dict → Contains scores for accuracy, latency, and clarity.
- benchmark_tests: list → A compilation of test cases for assessment.
- report_format: str → Specifies the format of the output (CSV, JSON, PDF).

Methods:

- run_performance_tests(test_data: Dataset) -> PerformanceResults → Assesses the effectiveness of Speech-to-Text (STT) and Text-to-Speech (TTS) systems.
- generate_report(format: str) -> ReportFile → Produces a report that encapsulates the performance metrics of the system.

3.6 API Integration & FNSS Compatibility

The API Integration & FNSS Compatibility module is essential for ensuring effective communication between system components and external services. It encompasses the following elements:

- External Service Communication: Guarantees compatibility with FNSS services and various third-party applications.
- Efficient Query Processing: Enhances API request handling to reduce response times and boost overall system performance.

- Standardized Protocols: Utilizes RESTful APIs and adheres to industry standards to ensure interoperability and reliability.

This module is pivotal in facilitating efficient system interoperability and secure data exchange.

Attributes:

- api_url: str → This parameter specifies the endpoint used for communication with external APIs.
- request_timeout: int → This parameter establishes the maximum duration to wait for responses from the API.

Methods:

- send_request(endpoint: str, payload: dict) -> ApiResponse → Transmits information to an external API.
- receive_response(response: ApiResponse) -> dict → Analyzes and processes the data received.
- optimize_query_performance(request: ApiRequest) -> OptimizedRequest → Enhances the efficiency of calls to external services.

4 Glossary

STT (Speech-to-Text): The technology that transforms spoken words into written text . Deep learning techniques are used by the Speech-to-Text component to interpret speech in real-time, assure transcription accuracy, and adjust to various dialects. It uses speech recognition models to convert characteristics into Turkish words, audio processing to reduce noise, feature extraction to find linguistic and phonetic components, and text refinement to assure fluency and fix mistakes.
TTS (Text-to-Speech): This technology ensures correct pronunciation, tone, and rhythm in Turkish by transforming written text into artificial speech. A preprocessing module, an acoustic model, a vocoder, and customization tools that let users change the volume, speed, and pitch are all included in the Text-to-Speech functionality.
NLP (Natural Language Processing): Making it possible for computers to comprehend and generate human language is the main goal of this branch of artificial intelligence.

FNSS: This refers to FNSS Savunma Sistemleri A.Ş., identified as an external partner that specializes in Defence Industry. The system is designed for compatibility with FNSS services.

API (Application Programming Interface): A set of features and protocols known as an API makes it easier for various software programs to communicate with one another. An API-based approach is used by the Turkish Language Processing System to provide seamless data exchange with FNSS services. By using encryption and authentication methods to protect transmitted data, the API Integration & FNSS Compatibility module is crucial for guaranteeing efficient connection between system components and external services.

UML (Unified Modeling Language): A standardized language for software design visualization. Class diagrams, sequence diagrams, component diagrams, and data flow diagrams are all created using UML diagrams in this project.

IEEE 830-1998: A standard that outlines the structure for documentation of software requirements. The functional, performance, and security requirements of the system are stated in accordance with this standard.

IEEE 1016-2009: A standard for software design documentation. By including the system architecture, database model, interface definitions, and control flow, the low-level design report complies with this standard.

GDPR (General Data Protection Regulation): A European Union law that regulates the gathering, use, and preservation of personal data. The project is set up in accordance with GDPR to protect the privacy and security of user data.

KVKK (Kişisel Verileri Koruma Kanunu): Turkey's personal data protection law that guarantees data security and privacy. By utilizing data encryption, authentication and authorization procedures, data retention guidelines, anomaly detection, and security monitoring systems, the project is in line with KVKK's efforts to safeguard user data.

JSON (JavaScript Object Notation): Information is structured and transmitted using this lightweight data interchange standard. The system exchanges structured data using JSON.

Deep Learning: A subfield of machine learning that uses artificial neural networks to find intricate patterns in data.

Transformer Model: A deep learning architecture intended for use in natural language processing and speech recognition applications. In an LLM context, transformer-based models are trained on dialect-specific datasets using unsupervised learning.

Benchmarking: Comparing a system's performance to predetermined measurements. The TTS and STT functions' efficacy can be measured and assessed using the tools in the Reporting & Benchmarking component.

RESTful API: An architectural method for developing networked applications that communicate via HTTP requests. To guarantee dependability and interoperability, the system uses RESTful APIs and complies with industry standards.

Acoustic Model: A deep learning model that links phonetic representations to acoustic characteristics. The acoustic model uses deep learning algorithms to generate speech parameters.

Vocoder: A voice synthesis component that converts acoustic data into real audio waveforms. The vocoder uses produced parameters to produce actual voice waves.

Signal Processing: A group of methods used to improve, examine, or work with audio signals. The Noise Reduction & Sound Isolation module improves speech clarity before transcription or synthesis by using advanced signal processing techniques.

Noise Filtering: The process of removing undesirable background noise in order to improve audio clarity is known as noise filtering. While preserving the integrity of the speech, the Noise Reduction & Sound Isolation module detects and removes background noise.

5 References

[1]: IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

[2]: IEEE 1016-2009 - IEEE Standard for Information Technology—System Design Documentation.

[3]: GDPR (General Data Protection Regulation), European Union Data Protection Law.

[4]: KVKK (Kişisel Verileri Koruma Kanunu), Personal Data Protection Law of Turkey.