Project Name: POC- Tuning Advisor Speech-to-Text API to evaluate no match utterances for problematic input states and generate tuning advice reports for improving the corresponding grammars using Artificial Intelligence and Machine learning.

Project Link: <https://github.com/andrewmmambo/Tuning-Advisor-for-Speech-to-text-API>

Developer: Marima Andrew Mambondiumwe.

Date: March 2021-August 2021

Technology stack/Tools used.

1. **Speech Input**:
   * **Tools**:
     + Python libraries: **SpeechRecognition** for processing audio input.
     + Audio processing frameworks: **LibROSA** for feature extraction.
   * **Technologies**:
     + RESTful APIs for receiving audio data.
     + Audio codecs for handling different audio formats.
2. **Speech Recognition Module**:
   * **Tools**:
     + Deep learning frameworks: TensorFlow for training and deploying speech recognition models.
   * **Technologies**:
     + Pre-trained models Google’s Speech-to-Text API for initial recognition.
3. **Grammar Evaluation Module**:
   * **Tools**:
     + Natural Language Processing (NLP) libraries NLTK for text analysis.
     + Finite state transducer (FST) libraries for grammar parsing.
   * **Technologies**:
     + Rule-based systems for grammar comparison.
4. **Problematic Input States Detection**:
   * **Tools**:
     + Machine learning frameworks for pattern recognition and anomaly detection: TensorFlow.
   * **Technologies**:
     + Statistical analysis techniques for identifying outliers and uncommon patterns.
5. **Tuning Advice Generation**:
   * **Tools**:
     + Machine learning algorithms for generating recommendations: decision trees.
     + Rule-based systems for defining tuning strategies.
   * **Technologies**:
     + Data visualization libraries: Matplotlib for presenting tuning advice.
6. **Reporting and Visualization**:
   * **Tools**:
     + Web frameworks: Flask for building reporting interfaces.
     + Data visualization tools: Tableau for creating interactive visualizations.
   * **Technologies**:
     + Frontend technologies HTML, CSS, and JavaScript for user interfaces.
7. **Feedback Loop**:
   * **Tools**:
     + Database systems: MongoDB for storing user feedback.
     + RESTful APIs for collecting feedback from users.
   * **Technologies**:
     + User authentication and authorization frameworks for managing user access.
8. **Deployment**:
   * **Tools**:
     + Containerization tools: Docker for packaging the application and its dependencies.
     + Orchestration tools: Kubernetes for managing containerized applications in production.
     + Cloud platforms: Google Cloud Platform for hosting the API.
   * **Technologies**:
     + Continuous Integration/Continuous Deployment (CI/CD) pipelines for automating deployment workflows. Use Atlassian’s suite of products.

**1. Introduction**

The Tuning Advisor Speech-to-Text API is designed to evaluate no-match utterances, identify problematic input states, and generate tuning advice reports for improving grammars used in speech recognition systems.

**2. Architecture Overview**

**+---------------------+ +------------------------+ +------------------------+**

**| | | | | |**

**| Speech Input | ------> | Speech Recognition | ------> | Grammar Evaluation |**

**| (Audio/Text) | | Module | | Module |**

**| | | | | |**

**+---------------------+ +------------------------+ +------------------------+**

**| |**

**| |**

**v v**

**+------------------------+ +------------------------+**

**| | | |**

**| Problematic Input | | Tuning Advice |**

**| States Detection | | Generation |**

**| | | |**

**+------------------------+ +------------------------+**

**| |**

**| |**

**v v**

**+------------------------+ +------------------------+**

**| | | |**

**| Reporting and | | Feedback Loop |**

**| Visualization | | |**

**| | | |**

**+------------------------+ +------------------------+**

**2.2 Component Descriptions**

1. **Speech Input**:
   * Accepts speech input in various formats (audio files, streaming audio) along with metadata.
2. **Speech Recognition Module**:
   * Transcribes the input speech into text using advanced algorithms and models.
3. **Grammar Evaluation Module**:
   * Compares transcribed text with existing grammars to identify mismatches and ambiguities.
4. **Problematic Input States Detection**:
   * Analyzes transcribed text to detect problematic input states.
5. **Tuning Advice Generation**:
   * Generates recommendations for improving grammars based on detected input states.
6. **Reporting and Visualization**:
   * Provides reports and visualizations of analysis results and tuning advice.
7. **Feedback Loop**:
   * Allows users to provide feedback on tuning advice and recommendations.

**3. Detailed Component Specifications**

**3.1 Speech Input**

* **Description**: Accepts speech input in various formats and metadata.
* **Interfaces**: REST API, SDKs.
* **Technologies**: HTTP, JSON, audio processing libraries.

**3.2 Speech Recognition Module**

* **Description**: Transcribes speech input into text.
* **Interfaces**: Internal API.
* **Technologies**: Speech recognition algorithms, deep learning models.

**3.3 Grammar Evaluation Module**

* **Description**: Compares transcribed text with existing grammars.
* **Interfaces**: Internal API.
* **Technologies**: Grammar parsing algorithms, natural language processing.

**3.4 Problematic Input States Detection**

* **Description**: Analyzes transcribed text to detect problematic input states.
* **Interfaces**: Internal API.
* **Technologies**: Text analysis algorithms, pattern recognition.

**3.5 Tuning Advice Generation**

* **Description**: Generates recommendations for improving grammars.
* **Interfaces**: Internal API.
* **Technologies**: Machine learning, rule-based systems.

**3.6 Reporting and Visualization**

* **Description**: Provides reports and visualizations of analysis results.
* **Interfaces**: Web UI, API.
* **Technologies**: Data visualization libraries, web frameworks.

**3.7 Feedback Loop**

* **Description**: Allows users to provide feedback on tuning advice.
* **Interfaces**: Web UI, API.
* **Technologies**: User feedback mechanisms, database.

**4. Deployment Considerations**

* **Scalability**: Utilize scalable infrastructure for handling large volumes of data.
* **Security**: Implement robust security measures to protect sensitive data.
* **Integration**: Provide APIs and SDKs for seamless integration with existing systems.
* **Performance**: Optimize algorithms and infrastructure for real-time or near-real-time processing.

**5. Results and Conclusion**

the reports generated reduced the time taken analyzing nomatch utterances by grammar expects by 65% from the period September 2021- December 2021. Hence the POC (proof of concept) was a success. By fostering language skills and communication capabilities, the API can contribute to the global competitiveness of US businesses and professionals. Improved language proficiency can facilitate international collaborations, trade partnerships, and cultural exchanges, strengthening the US position in the global economy.