Iteration 1 Documentation

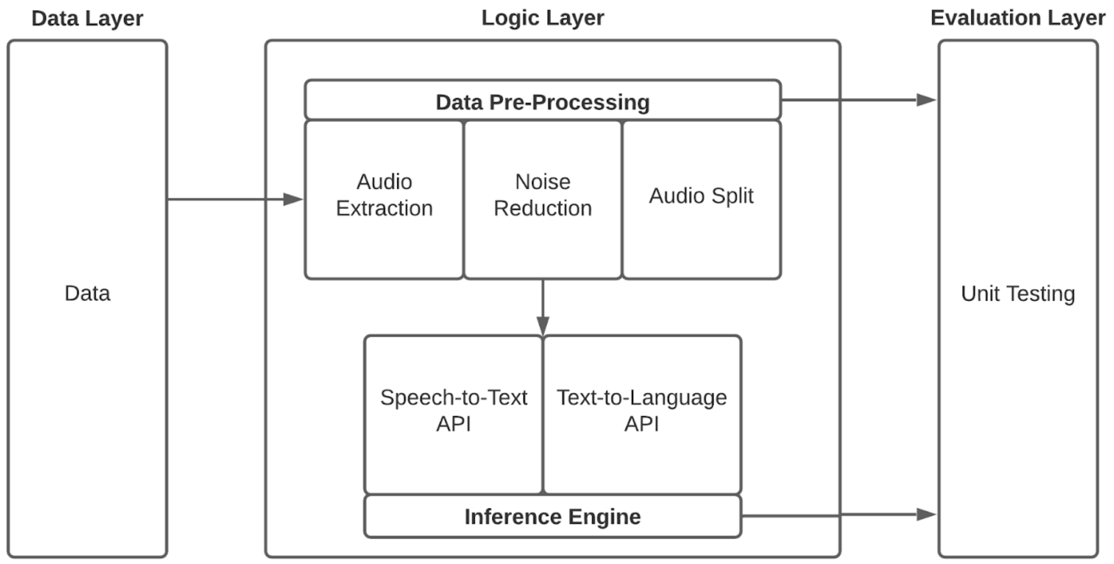
Customer meeting date to demo this iteration - April 20th

User stories implemented in this iteration -

Feature: Based on the feature selected, pre process the data. This includes cleaning the data, converting the raw audio/video file into the correct format for the ML model.

This story has been implemented for this iteration. Individual pieces of code have been developed and Unit testing has been done for them

Design diagram for this iteration:



Data Layer presents data to the logic layer. Logic layer has a pre-processing unit and Inference engine. Inference Engine has a model developed using Speech-to-Text API (Google Recognise API) and Text-to-language API (LangDetect). Then we perform unit testing on the model and data processing units.

1. Extract Audio from Video

   This code was implemented using the MoviePy library. It takes in a video file as input and extracts the audio stream from it and stores in the '.wav' format. We can optionally choose to specify the format of the desired audio file. The code was tested on the sample video files provided by the TBOP team. Additionally, a sample video file has been included in the GitHub repository along with the command in Readme to test the working of the code.

2. Split into 20 seconds

   This code was implemented using pydub from Audiosegment. It divides the audio file of wav. Format into small chunks of specified length. Pydub calculated in milliseconds. Thus a sample video of 1 minute was split into 3 chunks of 20 seconds each. The code was tried and tested on various audio files to determine its reliability

3. Remove Noise

   There are two ways to remove noise. One is by using the integrated features of Google API and mentioning an extra parameter in the next step.

   The other is a command utility called SoX (Sound Exchange). In SoX, we can apply a low pass filter to reduce all the background noise below a certain threshold. Silent parts of the audio can also be clipped by specifying a certain energy level below which the audio will be removed. You can visually see the difference by plotting a spectrogram.

4. Use Audio to text API

   The audio to text feature is implemented using the ‘Speech Recognition’ library. It provides us with a Recognizer class and each instance of this class offers us with seven methods for recognizing speech from an audio source (which is previously extracted from the classroom video recordings in our project). We are using Google Web Speech API for speech-to-text. The audio file is provided to the API in the wav format and the output is obtained in the form of text.

5. Use Text to Language API

   The implementation to predict the language of the text is done using Python language detection library - Langdetect. It has better accuracy and also the inference time is very fast. This library returns the detected language of longer sentences.
   To determine both english and spanish in the output text, we split each sentence and detect the language.

Custom Grading -

1. The code is written in python, therefore instead of cucumber and rspec tests individual code blocks are run and output is analysed.
2. The project would be based solely on a Machine Learning model to observe and analyze the data provided by the customer and aim to improve the performance

measurement **without the need** or requirement of any **User Interface.** The demo of the unit tests of user stories of iteration 1 will be provided to the customer in the next meeting on April 20th.

Github Repository: https://github.com/rjagiasi/TBOP
Pivotal tracker account: https://www.pivotaltracker.com/n/projects/2495399