# Feeble Audio Based Transcript Generation
# Course : IE 643

Guided by: Prof. Balamurugan.

TA's: Mr. Rahul, Mr. Bheeshm

Team Name: NLPeeps

Team Members:
Aryash Srivastava(22B1506), Dhruv Garg(22B1529)

# Outline

Through this presentation, we are going to explain the workflow/journey of our project.  In this presentation we will focus on:

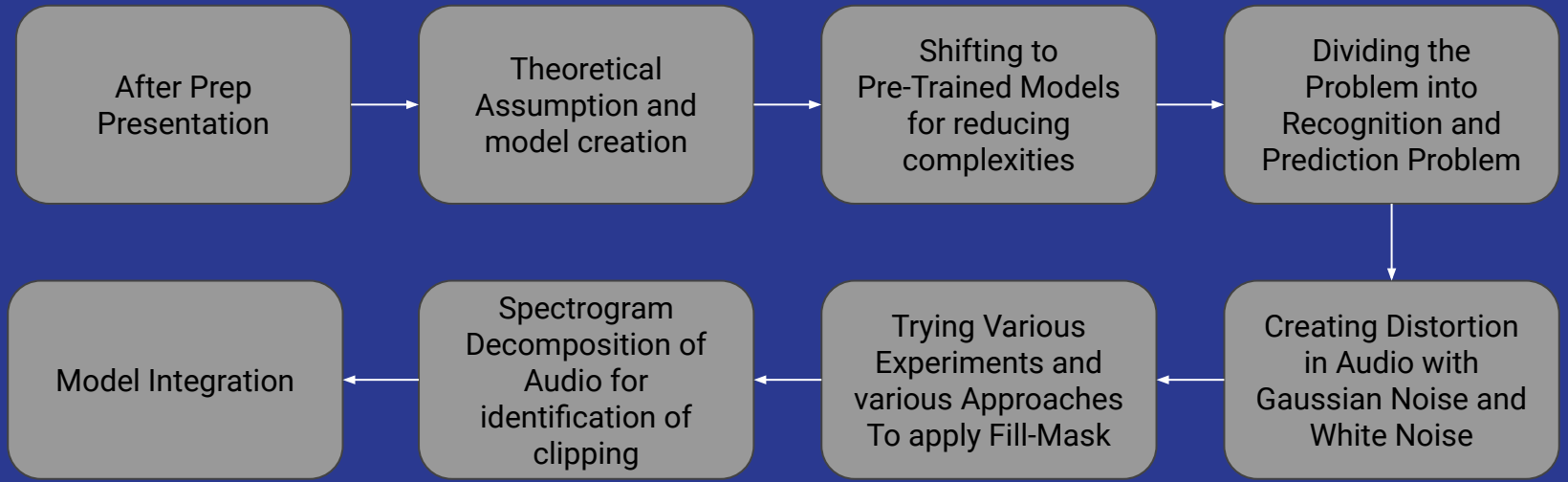
1. How we switched to pre-trained models
2. What all models we used
3. Algorithm behind our project
4. Implementation in Project
5. What all experiments failed and led to this solution

# Problem Statement:

Feeble audio based transcription generation with minimal syntax and information loss.

As we discussed with the Professor and TA’s we got to know that for feeble audio. We had to create a distorted noise and maybe generate audio which had a missing audio segment. Our Task is to provide clear and predict the whole text correctly.

# Workflow:


After Prep Presentation
Theoretical Assumption and model creation
Shifting to Pre-Trained Models for reducing complexities
Dividing the Problem into Recognition and Prediction Problem
Creating Distortion in Audio with Gaussian Noise and White Noise
Trying Various Experiments and various Approaches To apply Fill-Mask
Spectrogram Decomposition of Audio for identification of clipping
Model Integration

# Summary of Work Done Before Prep Presentation:

Work Done before Prep Presentation was totally Theoretical. Not Code was implemented. Initially, we thought to build Speech Recognition model from scratch which would help us learn about various other models such as Lexicon Models, Hidden Markov Model etc. The approach for prediction of segment was similar on the basis of n-grams. Where the mid-word will be predicted wihe the help of each word of sentence. We thought the process but did not implemented by writing any code.

# Description of work done after prep presentation review :

- Used Pre-Trained Model which helped us to focus on accuracy and approach
- Audio Distortion Generation with the help of White Noise and Clipping/Replacing the audio segments with pure noise.
- Working on Prediction
- Making the Machine Understand whenever there is a muted section using Spectrogram Decomposition
- Using Fill-Mask Hugging Face Model for prediction in that specific Region
- Model Integration

# Proposed Approach or Approaches:

There were a lot of approaches we tried but failed. We every time learned from our mistakes and tried to make it better always.

Problems:

1. Speech Recognition
2. Audio Distortion Generation
3. Muted Part prediction

# Proposed Approach or Approaches:

1. Speech Recognition

   Earlier we tried building the model. But due to increase in model complexity. We decided to go with pre-trained model.

2. Audio Distortion Generation

   For this we used White Noise using numpy random function. There were two parts:- Noise Addition, Segment replacement with Noise.

# Proposed Approach or Approaches:

3. Muted Part Prediction:

For this part, we were confused for a lot of time. Our main approach was:- Passing the Distorted Audio from the Speech Recognition Model. It gives some transcription which can be wrong due to muted sections. Now by iterating, we mask a specific word and we try to predict that area of the sentence using another all words. If the prediction shows that word then we can ensure there is nothing missing from that section of the sentence. Otherwise, we give our prediction which is added to the text.

Although this was a flaw method. Because there are multiple possibilities where word does not matches the predicted word. And we may end up with a lot of sections with predictions.

Alternate method try to find where the audio is muted and predict only for that part. This was done using spectrogram decomposition. Where while speaking there is a lot of variation in amplitude. However, during pure noise part the variance in amplitude is very low. This is how we made further predictions.

# Dataset and Data Preprocessing:

Dataset: The Mozilla "Common Voice" dataset from Kaggle consists of around 200,000 voice recordings ranging from 3 to 7 seconds along with corresponding text transcriptions.

Data Preprocessing : Clipping and Adding Noise using various libraries such as librosa to manage sampling rate. Adding a gaussian noise with amplitude randomly varying from 0 to 0.02. Replacing 0.2 to 0.5 seconds of audio with White Noise.

# Experimental Setup And Details:

Model Integration is the core part of the project where suddenly all of success and working of the project is visible.

Earlier different blocks of code were scattered all across of notebook. Such as spectrogram_function, add_noise_function,clip_audio_function, Predict_word_function, audio_breaking_function etc. Now it was time to integrate and wire all the components instead of linking everything manually.

different base models were created.
1) Distortion Generator
2) Differentiator{on the basis of spectral analysis which tells us how to break sentence}
3) Prediction Function

# Experimental Setup And Details:

All these Base Models were linked further to bring up the “Final_Predictor”:

```
def Final_Predictor(audio_file_path):
    noise_output_path=Base_Distortion_Creator(audio_file_path)
    masked1,masked2=Base_Differentiator_Model(noise_output_path)
    predictor_of_masked_sentences(masked1,masked2)
```

A lot of task was finished. Since we just had to pass audio_file_path to final_predictor and it would automatically add noise to clear input and predict the actual output from the distorted noise.

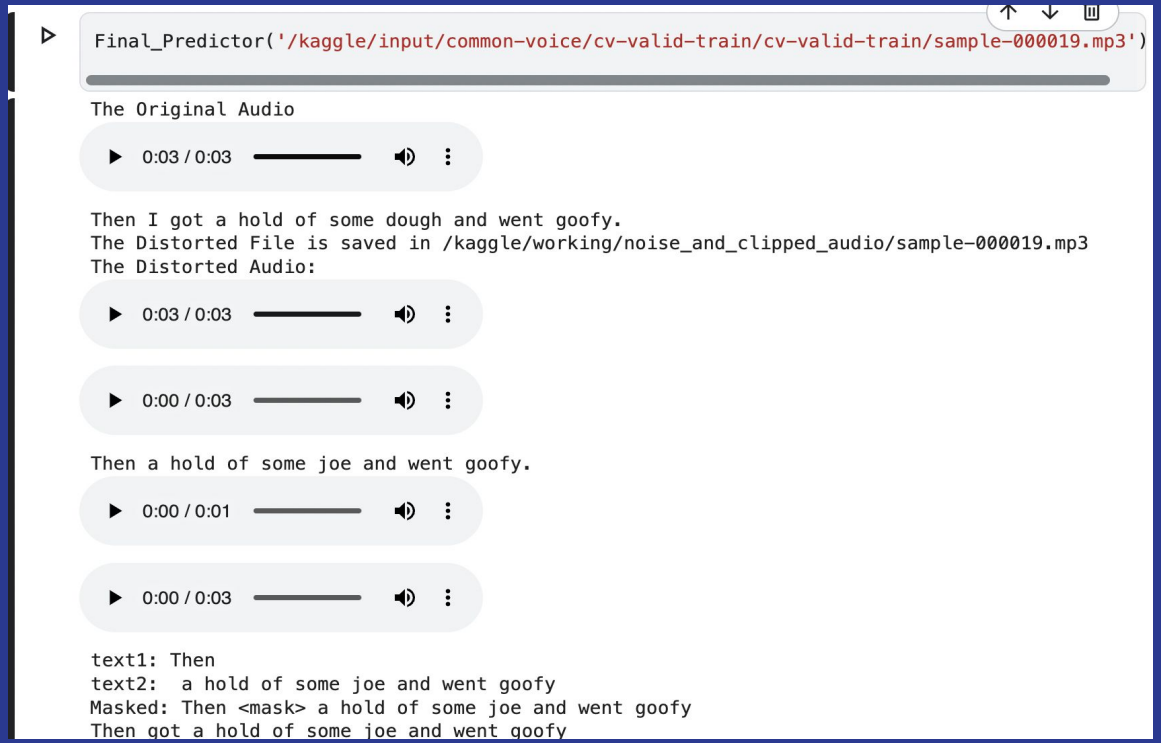
# Experimental Setup And Details:

Test_Predictor was something which was used for real_time testing.

```
def Test_Predictor(audio_file_path):
    masked1,masked2=Base_Differentiator_Model(audio_file_path)
    predictor_of_masked_sentences(masked1,masked2)
```

As you can see it can be easily created by using those 2 base models . 1) Base_Differentiator_Model for detecting muted section and 2)predictor_masked_sentences to predict the in-between word.

# Experimental Results Dataset Testing :




Actual Statement:

“Then I got a hold of some dough and went goofy.”

Machines Understand where there is a muted section:

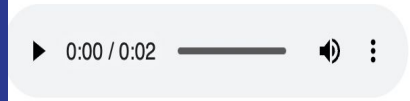
“Then <mask> a hold of some joe and went goofy.”

Predicted Statement:

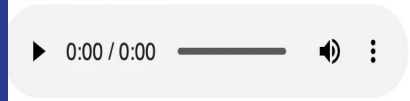
“Then got a hold of some joe and went goofy.”

# Experimental Results Real Time:

```
Test_Predictor('/kaggle/input/real-time5/Record (online-voice-recorder.com) (4).mp3')
```




```
Sun from East.
```





```
text1: Sun
text2:  from East
Masked: Sun <mask> from east
Sunrise from east
```

Real Time Audio {recorded from voice recorder}

Machines Learn the area which can be muted: and declares a masked region

Real Sentence{errors}: Sun From East

Predicted: Sun Rises from East

# Conclusions:

Hence, by Experimental results we can see that. It is not 100% accurate but it predicts the text almost correctly in most of the cases. Performance metric can not be used here, because even similar words will cover same semantic meaning but will not be classified same.

# Novelty Assessments:

First try to find all the mistakes: such as multiple mutes section creates a lot of problems. Improving the quality. And Then we can think of deployment of the project.

# References:

Just taking the idea or concept:

https://courses.grainger.illinois.edu/ece590sic/sp2024/SpecAugment.pdf

https://ar5iv.labs.arxiv.org/html/1810.04826