Natural Language Processing

Project Increment 1

Project Description:

The introduction should be composed of the following

1.

1.1 Project Title:

Text Summarization from Real Time Speech Recognition

1.2 Team Members:

Achala Samudrala –11519259 (Team Lead)

Pavani Mangugari –11542749

Tharun Puri – 11558533

Venkata Sai Preetham Bonthala – 11550407

1.3 GitHub Link:

https://github.com/PavaniMangugari/NLPProject-9

2. Goals and Objectives:

2.1 Motivation:

Information retrieval is one of the most widely needed use cases these days. The process of extracting the summary of a video lecture while remembering it and manually writing a summary is a time taking and intensive task for humans. There are plenty of text and video material available on the Internet. The process of developing a model for the machine to automatically convert the speech in the video to text would help the students and lectures.

Speech recognition would mean converting the speech to text in the real time scenario i.e., exactly converting the words with means of pronunciation. Text summarization is defined as the process of finding the meanings of difficult words and gets summarized to the shorter version where we can find them easily. So, we plan to develop a model that would take the lecture and interpret and summarize it. Additionally, we are using SR with text summarization to get Question and Answers that would help the students review their knowledge regarding the lecture

and automatically check the correctness. We also use Machine Translation to translate the document into another native language.

2.2 Significance:

In this age where remote education and online videos are becoming hugely popular, the need for a model that can convert speech to text is necessary. The model would not only help a lot of users, having problems with understanding the language, it would also help the users with disabilities too. The whole text and lecture are hard to memorize, so having a summary that would brief about the lecture would help the users while revising or just to know the undergoing of the lecture.

With the implementation of additional functionality, it will allow the users to view a few questions based on the lecture so that they can test their understanding in the lecture. This project aims to ease online learning and help users who are struggling whilst listening to these classes and get concise overview of the lecture in their native language.

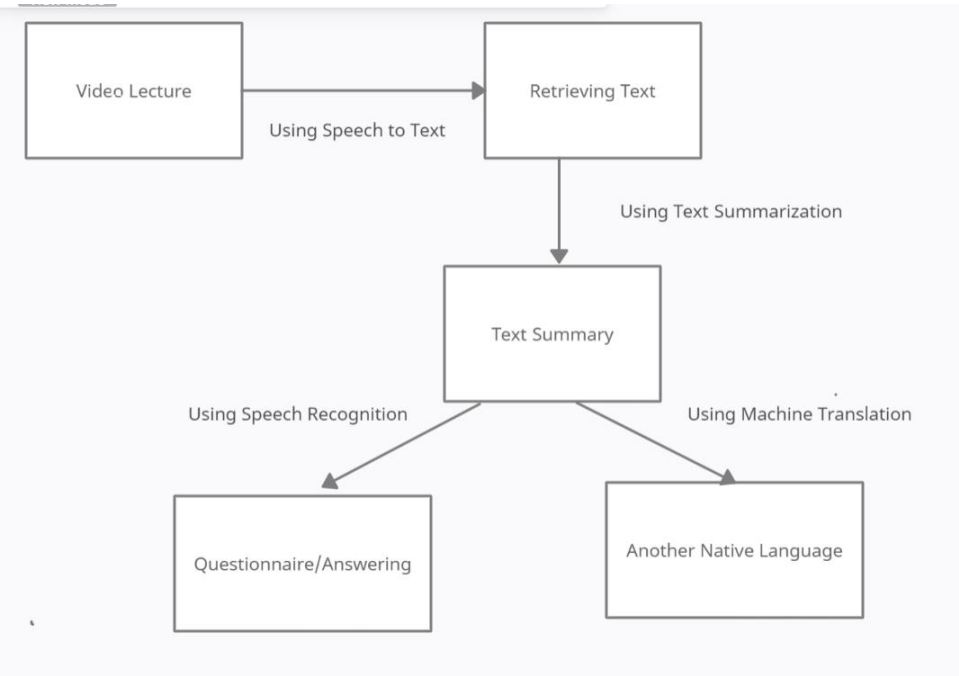
2.3 Objectives:

We are going to implement this project in 3 steps.

➢ Speech to Text

➢ Text Summarization

➢ Q&A module.



We are planning to do the project using the speech to text, text summarization and other training models. We will use the NLTK, SPACY, Genism, and Hugging Face tools to accomplish our project. We use machine translation for translating the text to another language.

We are at the end of the project planning to achieve an interface that would take the video input that converts it to text summary. In turn it uses this summary to build a Q/A model and Machine Translator for translating it to another language (Spanish/French). The goal of our project is to develop a model that will be accurate and precise in the summarization and processing that would help the users.

2.4 Features:

The video speech is converted into text using the python modules that would allow for the speech to turn into the text. We retrieve the text, perform the text cleaning and processing. We then tokenize the words, perform summarization techniques to get a video summary.

Using the machine translation, we convert the text into French or Spanish. We get the questions and answers from the summary. The user would be allowed to click for a correct answer. The answer would then be evaluated by the machine and returns if the user is right or wrong.

We will be implementing the feature that the user will be clicking on the appropriate answer so that he can check the answer whether it is correct or not. `

Increment 1 Guidelines

1. Related Work (Background)

Paper-1: They have considered that the system should consider the word by word from multiple speakers at a time so that it can identify the number of speakers. They have used some profiling techniques to reduce the signal-noise ratio by means of normalization and splitting the words so that they can get the ratio of 1:1. They got the accuracy of 71.7%.

Paper-2: In this they have taken the isolated word recognition and continuous phoneme recognition in a cued speech. They have concatenated the lip movement and hand shape for feature fusion and HMM based recognition has been implemented. They got 94% and 89% as accuracy for normal hearing and deaf people hearing.

Paper-3: In this paper they have used Devanagari script for classifying their classes based on phonemes. It converts the speech to phonemes by means of simple operations like zero-crossing and FTT. They did not get satisfactory results as the vowels and other consonants overlap in the same group. They got the accuracy of 75%, which is not satisfactory.

Paper-4: In this they have converted the normal speech to whispered speech by means of signal processing and voice conversion techniques. They have used gaussian mixture model (GMM) and deep neural networks (DNN) to map the features. They have also evaluated the naturalness and the speaker similarity after converting them to whispered speech. However, the model has failed in detecting the gender based on the speaker.

Paper-5: They have taken the double handed Indian sign language and converted them into patterns and converts speech to speech less. For this they have used the clustering techniques which may be useful for classifying the text and minimum eigen value algorithm for conversion. They have used bare hands for sign language rather than gloves. They have converted sign language into both text and speech.

Paper-6: they have taken both speech and object detection for speech recognition. They have taken the cognitive model for object detection based on speech to text recognition. In This they have developed a speech command and given it as input and by that the object is detected. The object is detected within a square box. For speech they have used google speech recognition and for object detection they have used YOLO V3 algorithm for that.

Paper-7: Here they are converting text to speech by mapping the source features. They have proposed a novel architecture used in Deep Learning. By considering the linear constraints, it prevents the input from converting features into unrealistic features. It is done by network-based conversion in a time variant way. It is suitable for multiple frames features.

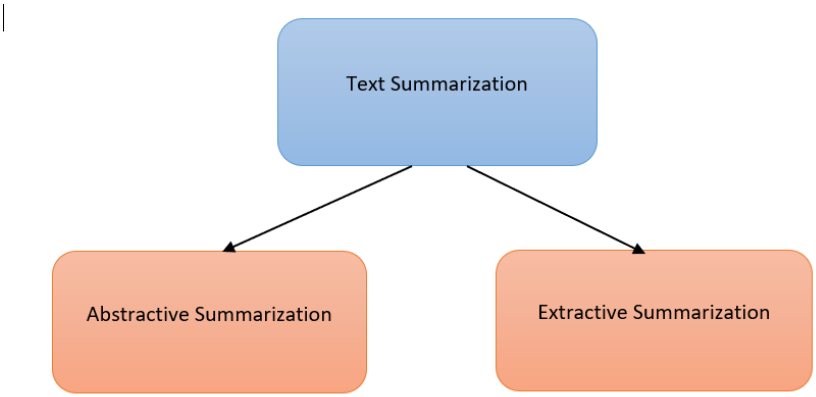
2. Dataset

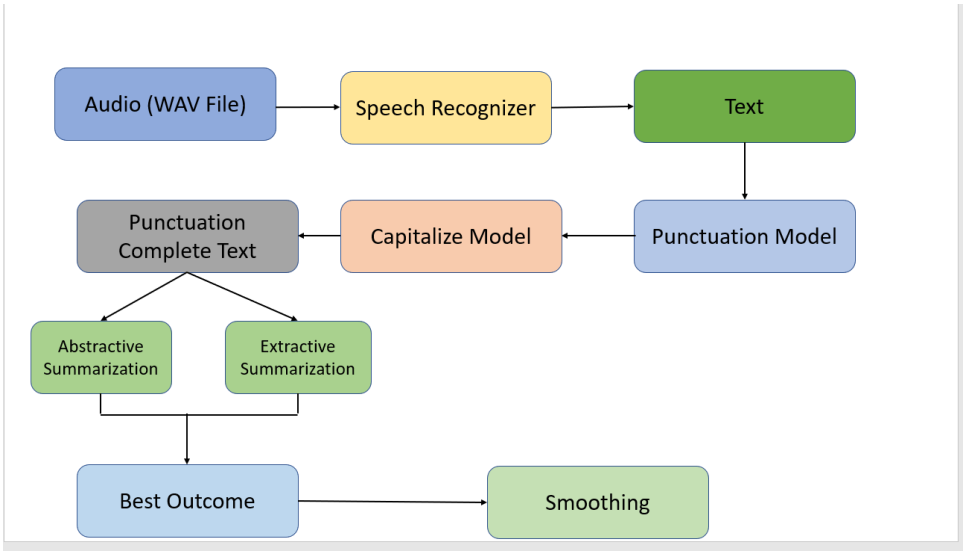
In the process of developing the application, we send audio data in the wav format. The application can handle data of a long duration. We tested the dataset of a university lecture about food processing. It is one hour long and has a lecture where there is one speaker.

Using the wave, we find the features of our dataset. We can see that we have 2 channels, with a sample width of 2. The frame rate is 44100 and the number of frames is 175584384.

```
import wave
obj = wave.open('food.wav','r')
print( "Number of channels",obj.getnchannels())
print ( "Sample width",obj.getsampwidth())
print ( "Frame rate.",obj.getframerate())
print ("Number of frames",obj.getnframes())
print ( "parameters:",obj.getparams())
obj.close()
```

```
Number of channels 2
Sample width 2
Frame rate. 44100
Number of frames 175584384
```





3. Detail design of Features

Our application first takes an audio input as a wav file

|

The audio file is first processed using the wave libraries, the matplotlib and get graphs regarding the audio frequency vs time.

|

Later, we take the text and pass it through a speech recognizer where the speech is converted into the text

|

The processed text is passed through a punctuationModel() where the information that we get from the recognizer.

|

The information that we get from the punctuation model is then sent and we capitalize words using capitalize() method.

|

Now, we get the text from the reformed model and perform text summarization using two different methods

|

Finally, we use the bigram model on the summarized text and apply Laplace and interpolation smoothening techniques and find accuracy.

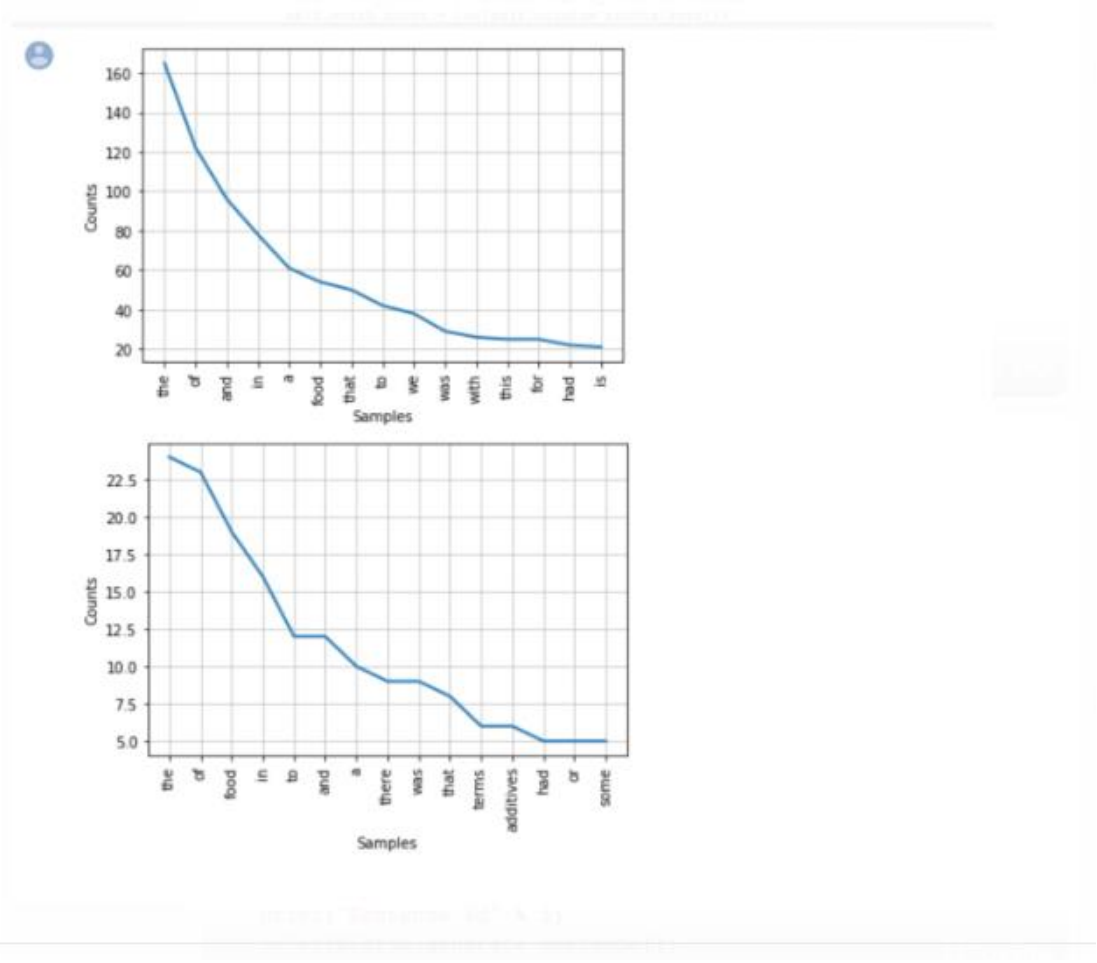
4. Analysis

In the analysis phase, we try to take the audio input that we have taken and try to analyze it. Later, we also try to analyze the differences in the word count in the original vs the summarized text in the summarizers that we have used.

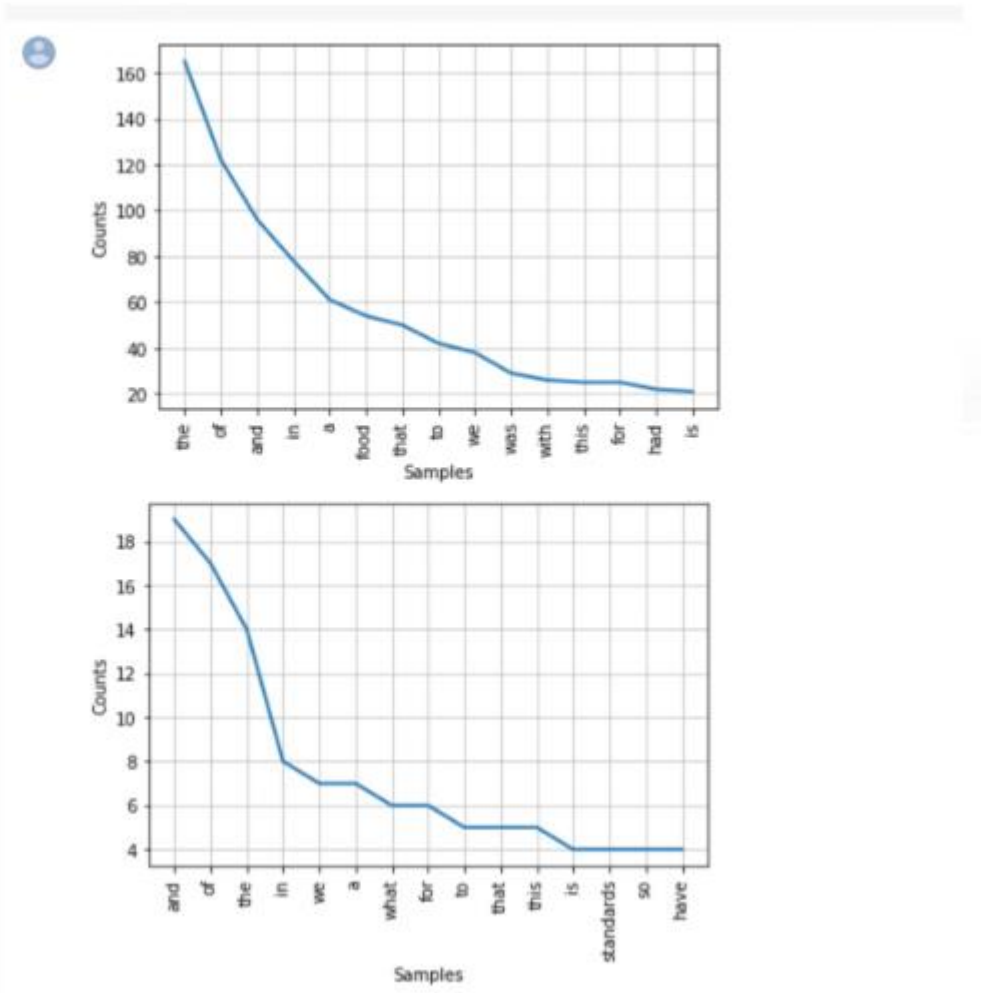
```
[41] import wave
     obj = wave.open('food.wav','r')
     print( "Number of channels",obj.getnchannels())
     print ( "Sample width",obj.getsampwidth())
     print ( "Frame rate.",obj.getframerate())
     print ("Number of frames",obj.getnframes())
     print ( "parameters:",obj.getparams())
     obj.close()

     Number of channels 2
     Sample width 2
     Frame rate. 44100
     Number of frames 175584384
     parameters: _wave_params(nchannels=2, sampwidth=2, framerate=44100, nframes=175584384, comptype='NONE', compname='not compressed')
```

We first analyze the audio dataset and can see the features of the dataset.

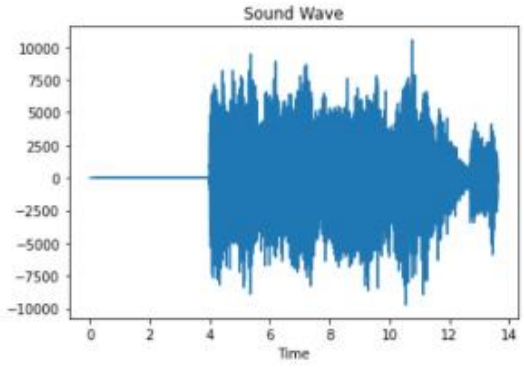


Difference between the counts in the abstractive summarizer vs the original text



Difference between the counts in the extractive summarizer vs the original text.

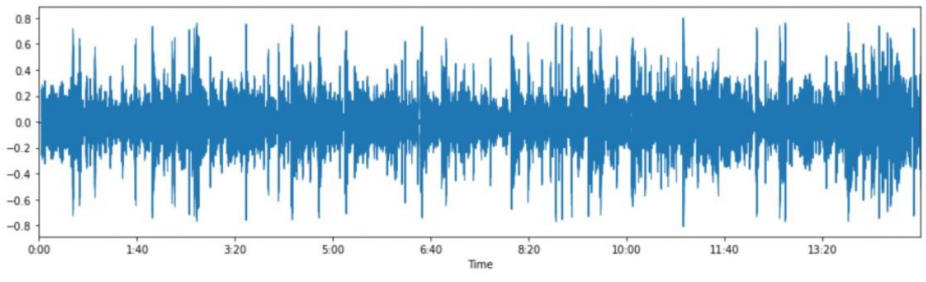
```
#opening the file using wave
raw = wave.open("food.wav")
  #reading the signal of 300000 frames
signal = raw.readframes(300000)
signal = np.frombuffer(signal, dtype ="int16")
  # gets the frame rate
f_rate = raw.getframerate()
  #for plotting the time, we divide the len of the signal with the frame rate.
time = np.linspace(
  0, # start
  len(signal) / f_rate,
  num = len(signal)
)
  #using matplotlib for the plot
plt.figure(1)
plt.title("Sound Wave")
plt.xlabel("Time")
plt.plot(time, signal)
plt.show()
```



Drawing a graph between the time and signal of the audio source.

```
#using the librosa library and plot the wave file for time and sample rate for the provided wav file.
plt.figure(figsize=(15,4))
data1,sample_rate1 = librosa.load('food.wav', sr=44100, mono=True, offset=0.0, duration=900, res_type='kaiser_best')
librosa.display.waveplot(data1,sr=sample_rate1, max_points=50000.0,x_axis='time', offset=0.0, max_sr=1000)
```

```
<matplotlib.collections.PolyCollection at 0x7f5aa4d41710>
```



A graph between the time duration and the energy involved which is the sampling rate and the signal.

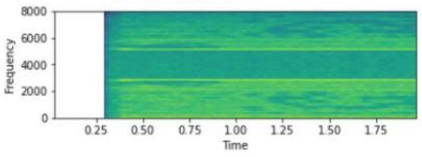
```
#using the wav file to plot the energry vs sample rate and time vs frequency graphs

signal_wave = wave.open("food.wav", 'r')
sig = np.frombuffer(signal_wave.readframes(16000), dtype=np.int16)
plt.figure(1)


plot_b = plt.subplot(212)
plot_b.specgram(sig, NFFT=1024, Fs=16000, noverlap=900)
plot_b.set_xlabel('Time')
plot_b.set_ylabel('Frequency')

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/axes/_axes.py:7592: RuntimeWarning: divide by zero encountered in log10
  Z = 10. * np.log10(spec)
```



The analysis between the time and the frequency of the audio wav file.

5. Implementation

```
pip install SpeechRecognition
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: SpeechRecognition in /usr/local/lib/python3.7/dist-packages (3.8.1)
```

```
[4] import os
    import speech_recognition as sr
```

```
[5] r = sr.Recognizer()
    audio = sr.AudioFile('food.wav')
```

Using the speech recognizer module for reading the audio file.

Speech to Text

```
#intializing text to empty first
finalText=""
#the intial offset is 0
offset=0
#i in the range of for the duration offset, we repeat the conversion
for i in range(10):
  #using the recogniser to find the text from the audio
  r = sr.Recognizer()
  audio = sr.AudioFile('food.wav')
  with audio as source:
    audio = r.record(source, duration=300,offset=offset)
  # set offset as 300
  offset=offset+300
  text=r.recognize_google(audio)
  finalText=finalText+text ## recognizing the speech and converting into text
print(finalText)
```

```
hello my name is food toxicology would like to welcome back once again today the subject of the lecture is going to be the history of us food regulation and
```

Using the speech recognition model, we get converted text

```
from deepmultilingualpunctuation import PunctuationModel
!pip install transformers

model = PunctuationModel()
text = text1
clean_text=model.preprocess(text)
labeled_words=model.predict(clean_text)
result = model.restore_punctuation(text)
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: transformers in /usr/local/lib/python3.7/dist-packages (4.24.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: huggingface-hub<1.0,>=0.10.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (0.10.1)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4.13.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-packages (from transformers) (1.21.6)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.8.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.7/dist-packages (from transformers) (0.13.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-packages (from transformers) (6.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0,>=0.10.0->transformers) (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers) (3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages (from importlib-metadata->transformers) (3.10.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.25.11)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2022.9.24)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages (from requests->transformers) (2.10)
/usr/local/lib/python3.7/dist-packages/transformers/pipelines/token_classification.py:136: UserWarning: `grouped_entities` is deprecated and will be removed
  "`grouped_entities` is deprecated and will be removed in version v5.0.0, defaulted to"
```

```
[ ] #predicted accuracy
    print(labeled_words)
```

```
[['hello', ',', 0.89068866], ['my', '0', 0.9999896], ['name', '0', 0.9998072], ['is', '0', 0.9979177], ['food', '0', 0.9322935], ['toxicology', '0', 0.555714
```

Using the word punctuation model, we get the converted text with punctuations

Capitalizing Text

```
[ ] def capitalizng_sentence(data):
      sent_tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
      sentences_from_summary = sent_tokenizer.tokenize(data)
      corrected_sentences = [sentence.capitalize() for sentence in sentences_from_summary]
      print(' '.join(corrected_sentences))
```

```
[ ] capitalizedres=capitalizng_sentence(result)
    print(capitalizedres)
```

```
Hello, my name is food toxicology would like to welcome back once again. Today. The subject of the lecture is going to be the history of us food regulation,
None
```

Using the capitalizing text, we capitalize each sentence.

```
def summarize(text, per):
    nlp = spacy.load('en_core_web_sm')
    doc= nlp(text)
    tokens=[token.text for token in doc]
    word_frequencies={}
    for word in doc:
        if word.text.lower() not in list(STOP_WORDS):
            if word.text.lower() not in punctuation:
                if word.text not in word_frequencies.keys():
                    word_frequencies[word.text] = 1
                else:
                    word_frequencies[word.text] += 1
    max_frequency=max(word_frequencies.values())
    for word in word_frequencies.keys():
        word_frequencies[word]=word_frequencies[word]/max_frequency
    sentence_tokens= [sent for sent in doc.sents]
    sentence_scores = {}
    for sent in sentence_tokens:
        for word in sent:
            if word.text.lower() in word_frequencies.keys():
                if sent not in sentence_scores.keys():
                    sentence_scores[sent]=word_frequencies[word.text.lower()]
                else:
                    sentence_scores[sent]+=word_frequencies[word.text.lower()]
    select_length=int(len(sentence_tokens)*per)
    summary=nlargest(select_length, sentence_scores,key=sentence_scores.get)
    final_summary=[word.text for word in summary]
    summary=''.join(final_summary)
    return summary
```

```
[ ] text2=summarize(text1, 0.07)
    print(text2)

    so in fact they had to prove certain health benefits and so the whole idea of patent or quack medicines went away fairly rapidly after that or at least had t
```

We summarize text using the abstractive summarization

▾ Text Summarization(Extractive)

```
!pip install -q bert-extractive-summarizer
!pip install -q spacy
!pip install -q transformers
!pip install -q neuralcoref
```

```
[ ] from summarizer import Summarizer
    from pprint import pprint
```

```
[ ] model = Summarizer()

    Some weights of the model checkpoint at bert-large-uncased were not used when initializing BertModel: ['cls.predictions.transform.LayerNorm.bias', 'cls.predi
    - This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a
    - This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequen
```

```
[ ] full=""
    res = model(result)
    full = ''.join(res)
```

```
[ ] print(full)

    hello, my name is food toxicology would like to welcome back once again. one of the answers, one of the responses is to establish standards of behavior, stan
```

Using extractive summarization, we get the summarized text.

Bigram model and smoothening

```
from collections import defaultdict
from collections import Counter
from numpy.random import choice
from tqdm import tqdm

class Bigram():
    def __init__(self):
        self.bigram_counts = defaultdict(Counter)
        self.unigram_counts = Counter()
        self.context = defaultdict(Counter)
        self.start_count = 0
        self.token_count = 0
        self.vocab_count = 0

    def convert_sentence(self, sentence):
        return ["<s>"] + [w.lower() for w in sentence] + ["</s>"]

    def get_counts(self, sentences):
        # collect unigram counts
        for sentence in sentences:
            sentence = self.convert_sentence(sentence)

            for word in sentence[1:]:  # from 1, because we don't need the <s> token
                self.unigram_counts[word] += 1

            self.start_count += 1
```

```
        # collect bigram counts
        for sentence in sentences:
            sentence = self.convert_sentence(sentence)
            bigram_list = zip(sentence[:-1], sentence[1:])
            for bigram in bigram_list:
                self.bigram_counts[bigram[0]][bigram[1]] += 1
                self.context[bigram[1]][bigram[0]] += 1
        self.token_count = sum(self.unigram_counts.values())
        self.vocab_count = len(self.unigram_counts.keys())

    def generate_sentence(self):
        current_word = "<s>"
        sentence = [current_word]
        while current_word != "</s>":
            prev_word = current_word
            prev_word_counts = self.bigram_counts[prev_word]
            # obtain bigram probability distribution given the previous word
            bigram_probs = []
            total_counts = float(sum(prev_word_counts.values()))
            for word in prev_word_counts:
                bigram_probs.append(prev_word_counts[word] / total_counts)
            # sample the next word
            current_word = choice(list(prev_word_counts.keys()), p=bigram_probs)
            sentence.append(current_word)

        sentence = " ".join(sentence[1:-1])
        return sentence
```

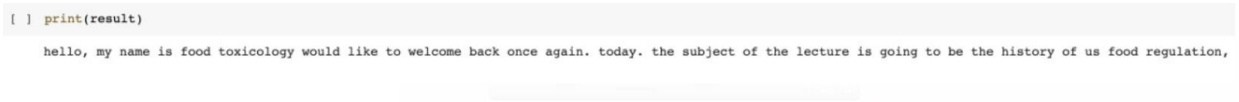
```
import nltk
from nltk.tokenize import sent_tokenize
##from nltk.corpus import brown
nltk.download('punkt')
s=sent_tokenize(result)
l=[]
for i in s:
  l1=list(i.split(' '))
  l.append(l1)

print(l)
bigram = Bigram()
bigram.get_counts(l)
for i in range(len(l)):
    print("Sentence %d" % i)
    print(bigram.generate_sentence())
```

We used the bigram model to generate the sentence from the summarized text.

6. Preliminary Results

First we get the text from converting the speech which is wav file.



Second, we add the punctuation to the text.

Then, we capitalize the each sentence after adding punctuations.

```
def interpolation(text2, bigram, lambdas):
    bigram_lambda = lambdas[0]
    unigram_lambda = lambdas[1]
    zerogram_lambda = 1 - lambdas[0] - lambdas[1]

    sentence = bigram.convert_sentence(text2)
    bigram_list = zip(sentence[:-1], sentence[1:])
    prob = 0

    for prev_word, word in bigram_list:
        # bigram probability
        sm_bigram_counts = bigram.bigram_counts[prev_word][word]
        if sm_bigram_counts == 0: interp_bigram_counts = 0
        else:
            if prev_word == "<s>": u_counts = bigram.start_count
            else: u_counts = bigram.unigram_counts[prev_word]
            interp_bigram_counts = sm_bigram_counts / float(u_counts) * bigram_lambda

        # unigram probability
        interp_unigram_counts = (bigram.unigram_counts[word] / bigram.token_count) * unigram_lambda

        # "zerogram" probability: this is to account for out-of-vocabulary words, this is just 1 / |V|
        vocab_size = len(bigram.unigram_counts)
        interp_zerogram_counts = (1 / float(vocab_size)) * zerogram_lambda

        prob += math.log(interp_bigram_counts + interp_unigram_counts + interp_zerogram_counts)
    return prob

bigram_interpolation = Bigram()
bigram_interpolation.get_counts(training_set)
plex_interpolation = calculate_perplexity(test_set, bigram_interpolation, interpolation, (0.8, 0.19))
print(plex_interpolation)
```

```
100%|██████████| 19/19 [00:00<00:00, 4788.31it/s]1070.463080935843
```

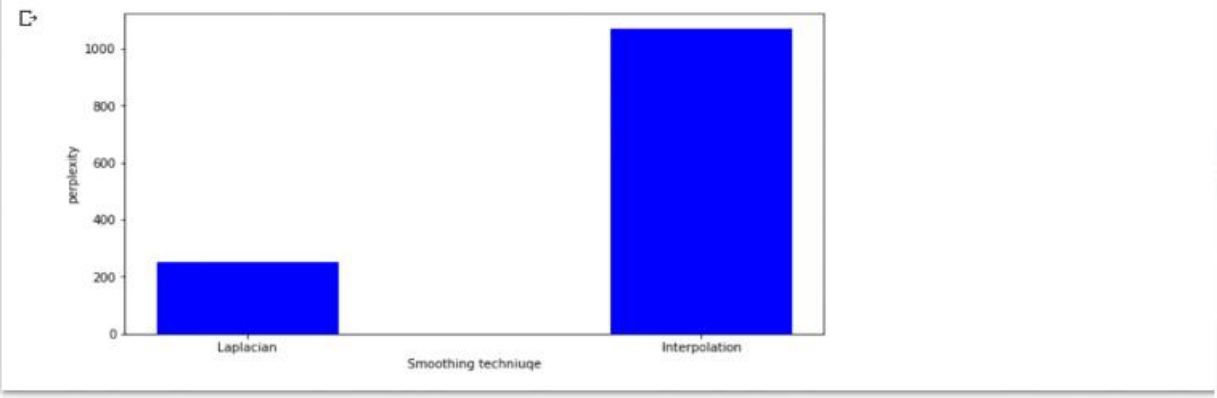
```
import numpy as np
import matplotlib.pyplot as plt

data={'Laplacian': 250.587,'Interpolation':1070.463}
ST=list(data.keys())
perp=list(data.values())

fig=plt.figure(figsize=(10,5))

plt.bar(ST,perp, color='blue', width=0.4)

plt.xlabel("Smoothing techniuqe")
plt.ylabel("perplexity")
plt.show()
```



Using the two models i.e., Laplacian and interpolation, we got the perperxities as shown below.

7. Project Management

Getting the best outcomes as well as directing the project with all the available resources is a crucial task. We have incorporated users' and participants' perspectives into this application's data gathering and analysis.

The objective of the project is to deliver a finished product that meets the requirements of the outcome. The project's objective is frequently to modify that to be more effectively achieve the project's goals.

8.Implementation Status Report

8.1 **Work Completed:**

We have completed 55% of the work on the project. For now, we have done Data preprocessing, speech to text recognition, text summarization, adding punctuation, applying smoothing techniques using Bigram model. In data preprocessing we have can get the number of channels, sample width, frame rate, Number of frames and parameters that are required. In speech recognition we have used the audio () library and recognized the text. We have done the text summarization from the speech recognized text. We have added punctuation to the recognized text. We have also applied smoothing techniques like interpolation and LaPlace add 1 using the Bigram model. We got the perplexity of both smoothing techniques.

- **Responsibility and contributions:**

S.no	**Person**	**Task description**	**Contributions**
1.	Achala Samudrala	Converted Speech-text recognizer	
2.	Pavani Mangugari	Applied Smoothing techniques	
3.	Tharun Puri	Added punctuation to the recognized text	

4.	Venkata Sai Preetham B	Summarized text from recognized text	

8.2 **Work to be completed:**

We must do UI implementation for the speech to text recognition. We have to implement Question and Answer module for the summarized text so that we can answer the questions and we also implementing another feature like clicking on the wrong answer should give us the correct answer so that we can check whether the answer is correct or not. We need to implement removing of background noise module so that we can get noise free audio. We need to implement a translation module so that the summarized text can be translated into native languages.

- **Responsibility and contributions:**

S.no	**Person**	**Task description**	**Contributions**
1.	Achala Samudrala	Implementing UI design for speech to text recognition.	
2.	Pavani Mangugari	Question and Answer module and updating the feature that is suggested	
3.	Tharun Puri	Removing background noise in the speech	
4.	Venkata Sai Preetham B	Translation of summarized text into native languages	

- **Issues/Concerns:**

Difficulty in overcoming the errors faced by the summarizer and speech to text recognition. We went through lot of online sources and worked on resolving errors.

It became hard for us to communicate as we have different class schedules. However, we have managed the schedules so that we can communicate among ourselves.

9. References/Bibliography:

https://ieeexplore.ieee.org/document/9622156

https://paperswithcode.com/paper/predicting-subjective-features-from-questions

https://paperswithcode.com/paper/iit-uhh-at-semeval-2017-task-3-exploring

https://paperswithcode.com/paper/qdee-question-difficulty-and-expertise

https://ieeexplore.ieee.org/document/9622156

https://paperswithcode.com/paper/predicting-subjective-features-from-questions

https://paperswithcode.com/paper/iit-uhh-at-semeval-2017-task-3-exploring

https://paperswithcode.com/paper/qdee-question-difficulty-and-expertise

https://ieeexplore.ieee.org/document/8465680

https://ieeexplore.ieee.org/document/5898904

https://ieeexplore.ieee.org/document/9315985

https://ieeexplore.ieee.org/document/8282216