B9AI109_2022_TMD3

CA-1 Part 2

Sunil Gauda
ID: 10595858

Business Understanding

A speaker's audio contains many parameters like pitch, amplitude, sound, and tone, but differentiating the emotions of spoken words can be done quickly when it's converted to text for analysis, as it consumes less computation power and has many tools and methods to process raw text.

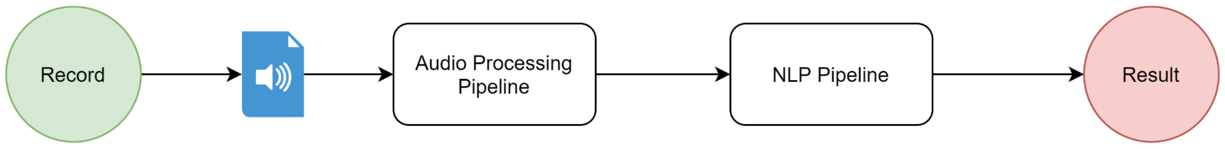


Fig 1 Audio Emotion Detection Pipeline

Above is the basic pipeline of the project, that is used to achieve the desired outcome. The emotion detection modal thus trained can be used in conjunction with a wide range of implementations from music recommendations to driver safety depending on the severity of the situation and accuracy of the model output. It can be used to measure anything from customer satisfaction to safety, with the addition of Smart devices in daily life it's easy to use such trained models to provide desirable outcomes.

Data Understanding

The Audio Data captured are live audio and comprise parameters that are as follows

Audio Parameters :

1. Chunk Size - It represents a sample size of the audio file in terms of frames to be accommodated in the memory
2. Format - Int is the audio format that we can use to analyze the audio recorded, using PyAudio provides us tools to read and analyze the data of each frame in numerical terms.
3. Channels - the number of points from which the sound will come out.
4. Rate - defines the speed at which the sound will pay, i.e number of frames per second.

When audio is recorded using the above variables required, the assigned values in the below audio wave diagram are as follows,

1. Chunk = 1024
2. Format = int (pyaudio enum)
3. Channels = 1
4. Rate = 44100

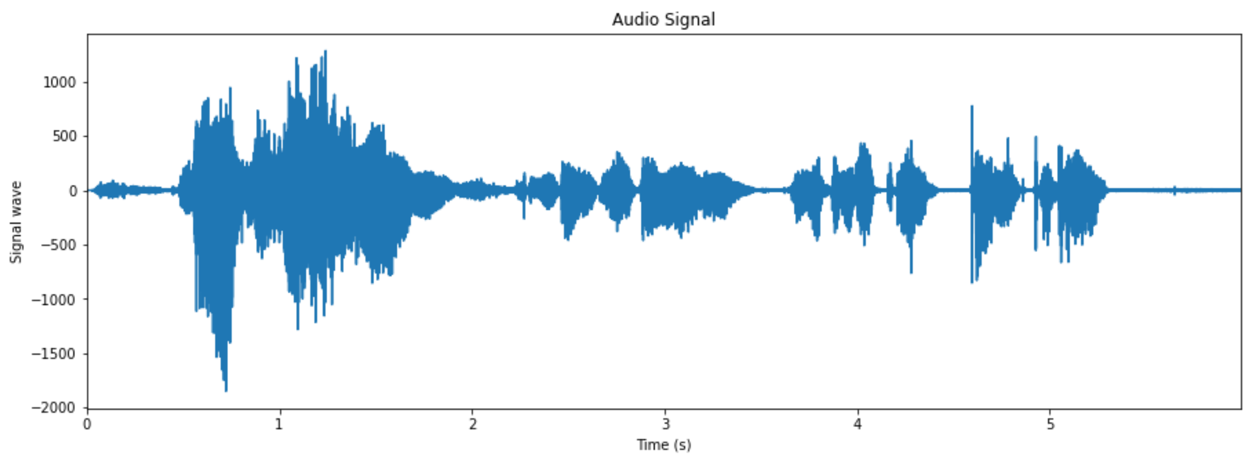


Fig 2 Visualization of an Audio

Data Preparation

Data Preparation includes the processing of pre-recorded audio the whole process of fetching pre-recorded audio and processing it is as follows.

1. The recording is done using PyAudio, a library available in python, please check the artifact for installation instructions.
2. Then the recorded audio is passed through an audio processing pipeline which works as follows.
    a. The recorded file is passed through the “speech_recognition” library which provides pre-trained deep learning models from providers like Google and Facebook, Google Speech To Text API is used here.
    b. We use Word Cloud to visualize Focus Words.

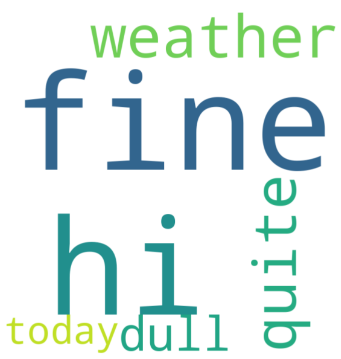
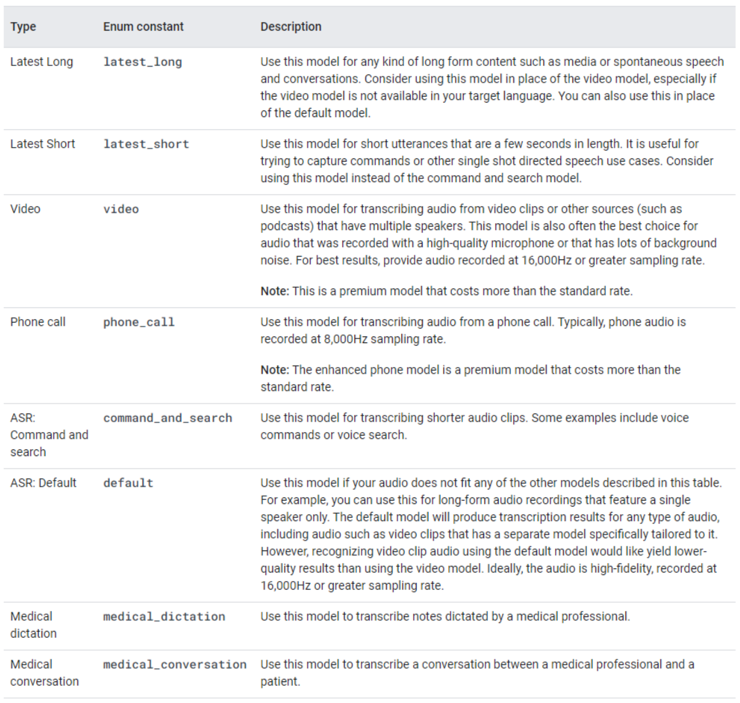


Fig 3 Word Cloud of spoken data

Unsupervised Learning with Google API

The API Consumes single 60-second audio and provides us with text data, It Unsperwised Recurring Network with LSTM as the model, it still has some features based on DNN models, but it is updated to suit the current models.



Type	Enum constant	Description
Latest Long	`latest_long`	Use this model for any kind of long form content such as media or spontaneous speech and conversations. Consider using this model in place of the video model, especially if the video model is not available in your target language. You can also use this in place of the default model.
Latest Short	`latest_short`	Use this model for short utterances that are a few seconds in length. It is useful for trying to capture commands or other single shot directed speech use cases. Consider using this model instead of the command and search model.
Video	`video`	Use this model for transcribing audio from video clips or other sources (such as podcasts) that have multiple speakers. This model is also often the best choice for audio that was recorded with a high-quality microphone or that has lots of background noise. For best results, provide audio recorded at 16,000Hz or greater sampling rate. **Note:** This is a premium model that costs more than the standard rate.
Phone call	`phone_call`	Use this model for transcribing audio from a phone call. Typically, phone audio is recorded at 8,000Hz sampling rate. **Note:** The enhanced phone model is a premium model that costs more than the standard rate.
ASR: Command and search	`command_and_search`	Use this model for transcribing shorter audio clips. Some examples include voice commands or voice search.
ASR: Default	`default`	Use this model if your audio does not fit any of the other models described in this table. For example, you can use this for long-form audio recordings that feature a single speaker only. The default model will produce transcription results for any type of audio, including audio such as video clips that has a separate model specifically tailored to it. However, recognizing video clip audio using the default model would like yield lower-quality results than using the video model. Ideally, the audio is high-fidelity, recorded at 16,000Hz or greater sampling rate.
Medical dictation	`medical_dictation`	Use this model to transcribe notes dictated by a medical professional.
Medical conversation	`medical_conversation`	Use this model to transcribe a conversation between a medical professional and a patient.

Fig 4 Variation of models offered by Google API

The First part of speech recognition of Google Speach to text is based on Unsupervised Context Learning for data less than 1 minute, which we have used in our **speech_recognition** API, The data set is our audio which is reduced, for conversion of speech to text latest_short, model is used, the model returns a text of the audio provided.

The Google Speech to text works using Deep Learning, It uses LSTM acoustic model with connectionist temporal classification (CTC) with SCD for convergence of data, the model is trained and refined by Google using huge datasets, and usually has one Language per model to improve accuracy and provide good results. Due to this google has a maximum of 13.5 percent of error in text prediction.

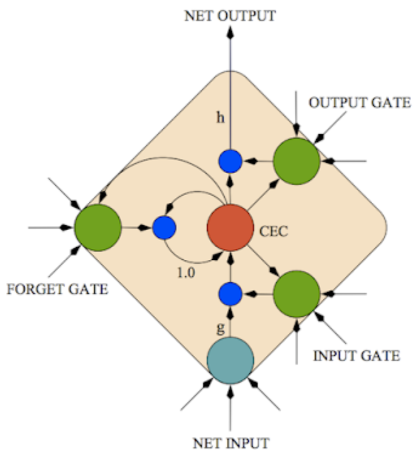


Fig 4 Google API Voice transcription

Modeling

As the dataset is small we have used Machine Learning Models to train and predict, the reference data are from yelp, amazon, and IMDB text datasets which are labeled and provide accurate output to train our model, as the goal is to detect emotions not just sentiments, the training data has emotions in it. We have used a Simple Liner Regression with Vectorisation to train and predict the outcome, the process is as follows.

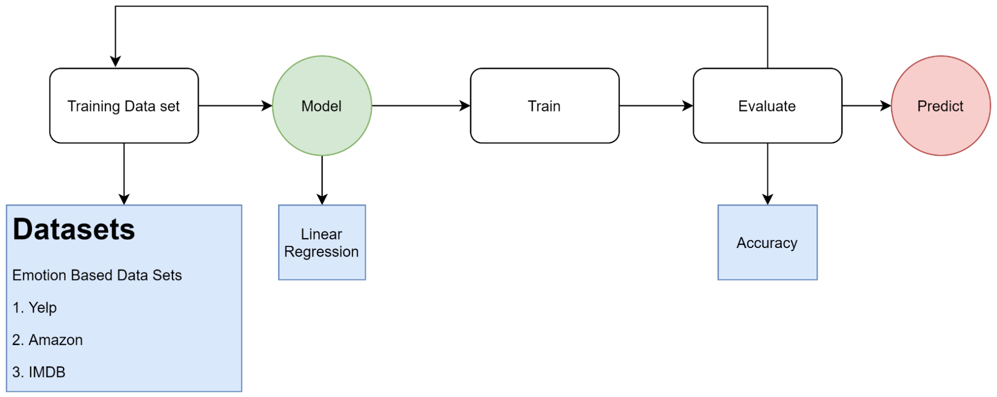


Fig 5 Basic Modelling Approach

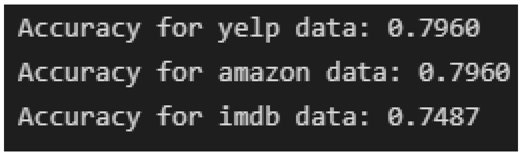
To feed the model the data we need to perform the following steps,

1. Instantiate the model, in this case, Linear regression
2. Load the data set to a data frame using pandas
3. Select the Feature and Input in this case “source” and “label”
4. Split the data set to train and test data
5. Use CountVectoriser to vectorize the data and feed it to model

Using the above model we can train on each data set available and pick the desired model after evaluation.

Evaluation

Accuracy is the evaluation parameter used, the accuracy of each dataset we have depends on the quality of text that represents the exact emotions we are looking for, We have used 3 datasets to check and evaluate model accuracy after training we get the following results.



```
Accuracy for yelp data: 0.7960
Accuracy for amazon data: 0.7960
Accuracy for imdb data: 0.7487
```

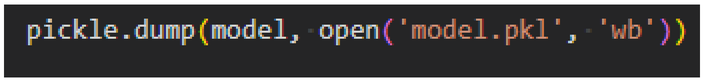
Fig 6 Model Accuracy

As we can observe that “yelp” and “amazon” data provide near the identical value of accuracy, the yelp dataset is used to train our final model. The final model thus has an accuracy that represents the exact state of emotions that is in the case of “Happy” or “Sad” the accuracy depends on the scale of data, as the speech data is small the accuracy of the final trained model might differ.

Deployment

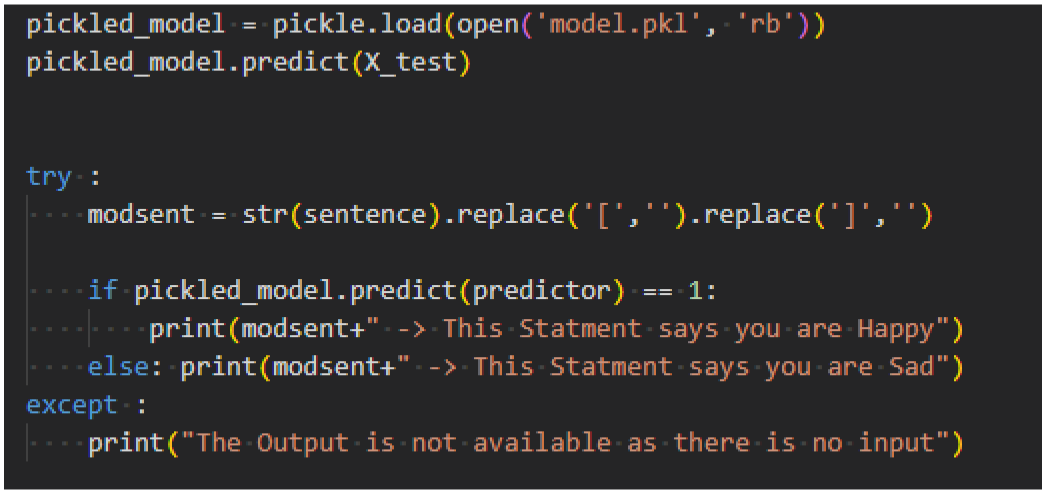
For deployment there are numerous options, as it is a machine learning model it is small and compact and can be pickled and stored using the pickle library in python and then it can be deployed and used in various applications, may that be web, desktop, or mobile.

The Process of storing and loading the file is as follows.



```
pickle.dump(model, open('model.pkl', 'wb'))
```

Fig 7 Pikle the model



```
pickled_model = pickle.load(open('model.pkl', 'rb'))
pickled_model.predict(X_test)


try :
    modsent = str(sentence).replace('[','').replace(']','')

    if pickled_model.predict(predictor) == 1:
        print(modsent+" -> This Statment says you are Happy")
    else: print(modsent+" -> This Statment says you are Sad")
except :
    print("The Output is not available as there is no input")
```

Fig 8 Loading and Running the model

References

1. Google AI Blog. (n.d.). *The neural networks behind Google Voice transcription*. [online] Available at: https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html.
2. Unsupervised Learning in LSTM Recurrent Network, CSE LAB, Harvard Available at: https://cse-lab.seas.harvard.edu/files/cse-lab/files/icann2001unsup.pdf
3. Michaely, A., Scheiner, J., Ghodsi, M., Aleksic, P. and Wu, Z. (2016). *Unsupervised Context Learning For Speech Recognition*. [online] Google Research. Available at: https://research.google/pubs/pub45759/ [Accessed 27 Jul. 2022].
4. McGraw, I., Prabhavalkar, R., Alvarez, R., Arenas, M.G., Rao, K., Rybach, D., Alsharif, O., Sak, H., Gruenstein, A., Beaufays, F. and Parada, C. (2016). *Personalized Speech Recognition On Mobile Devices*. [online] Google Research. Available at: https://research.google/pubs/pub44631/ [Accessed 27 Jul. 2022].
5. people.csail.mit.edu. (n.d.). *PyAudio Documentation — PyAudio 0.2.11 documentation*. [online] Available at: https://people.csail.mit.edu/hubert/pyaudio/docs/.
6. Zhang, A. (2022). *SpeechRecognition*. [online] GitHub. Available at: https://github.com/Uberi/speech_recognition [Accessed 27 Jul. 2022].
7. Wikipedia Contributors (2018). *Long short-term memory*. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Long_short-term_memory.