Yash Wali, Somesh Yadav,
Christoph Sieberer-Kefer

Project Report – Team Red Bull Racing

MSA8150 – Machine Learning for Analytics April 26th, 2022

Abstract

As part of our final project engagement, our group chose to analyze the “Topic Analysis” and “Voice Tone” datasets. The objective for the “Topic Analysis” project was to create a model that can accurately predict which topic a comment belongs to. We realized the importance of this project as companies such as social and news media rely on analyzing textual data to inform their decision-making process daily. Our solution can help them categorize comments more accurately, allowing them to analyze the sentiment of different topics and tailor their business solutions to identified needs. Our second project, “Voice Tone”, is interesting because there is not an abundance of pre-trained algorithms and models. Hence, we needed to empirically understand signal processing to process the audio data. This transformation can then be applied to extracting information from videos, for instance. Overall, our models were able to achieve accuracies of 82.6% and 50.3% for Response 1 and Response 2, respectively, for the "Voice Tone" project and an accuracy of ~37% for the "Topic Analysis" project.

Chapter 1: Topic Analysis

Introduction

Project Statement

Using a dataset of 900,000 comments belonging to one of 40 topics, we want to build a model that accurately predicts a comment’s topic by ways of extracting information from the text. Since the data is unstructured and does not follow an inherent pattern, the challenges will be to clean and process it thoroughly and allow our model to uncover hidden structures that determine the comments’ topic.

Metrics

The metric used to evaluate our model is accuracy which considers each correct prediction with an equal weight.

$$Accuracy = \frac{Correct\ predictions}{Total\ predictions}$$

Techniques Considered

In preprocessing the data and building our model, we considered a multitude of different techniques, each having their own advantages to increase the performance and leading to better results. The following preprocessing techniques were explored:

- To normalize inputs, we tried the "Snowball Stemmer" algorithm. The "Snowball Stemmer" is an improved version of the "Porter Stemmer" which is a process used to eliminate morphological and inflectional endings from words[1]. As opposed to lemmatization, stemming is a heuristic process that rids the word of its ending, often collapsing derivational affixes and disregarding the use of the word (e.g., as a noun or verb)[2], leading to less desirable outcomes.
- Reducing each word to its word stem using the "WordNetLemmatizer" algorithm. It is an algorithm that makes use of vocabularies and morphological analysis of words with the aim to remove inflectional endings from words[2]. This is done to normalize the inputs and reduce the variations and parameters our model will be trained on.

As part of our feature engineering process, we tested two different vectorization methods:

- Count based feature representation converts our collection of comments to a matrix of term (token) counts. It is a very crude way to encode words as numerical features by simply counting the number of occurrences across all documents.
- A more sophisticated way to represent each word is to calculate its term frequency but assign a penalty if it occurs across comments to reduce its weight. This reflects how important a word is to a comment in a collection of comments.

To produce the best accuracy, we implemented a number of different machine learning, deep learning, and transfer learning models. Each of the following has been tested on the same input:

- Random Forest (RF)
- Convolutional Neural Network (CNN)
- Long Short-Term Memory (LSTM)
- Bidirectional Encoder Representations from Transformers (BERT)

Apart from the two deep learning models (CNN and LSTM) who, contrary to our expectations, performed rather poorly, it was a close race between RF and BERT with BERT ultimately performing slightly better with respect to accuracy and runtime.

---

1 https://www.projectpro.io/recipes/what-is-difference-between-porter-and-snowball-stemmer

2 https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html#:~:text=Lemmatization%20usually%20refers%20to%20doing,is%20known%20as%20the%20lemma%20.

Problem Analysis

Data Exploration

The dataset at hand consists of 900,000 comments, each belonging to one class. The comments can range from a single word to over 1800 words and there are a total of zero null values prevalent in the dataset. The class distribution is ever so slightly unbalanced with the lowest count being 22436 and the highest being 22572, though this small discrepancy should not have too much of an impact on our model performance.

Further investigation into the dataset has unveil that the length of comments is positively skewed as shown in **Figure 1**. The bulk of comments is below 50 words in length with comments between 1 and 10 words accounting for over 40% of the entire dataset.

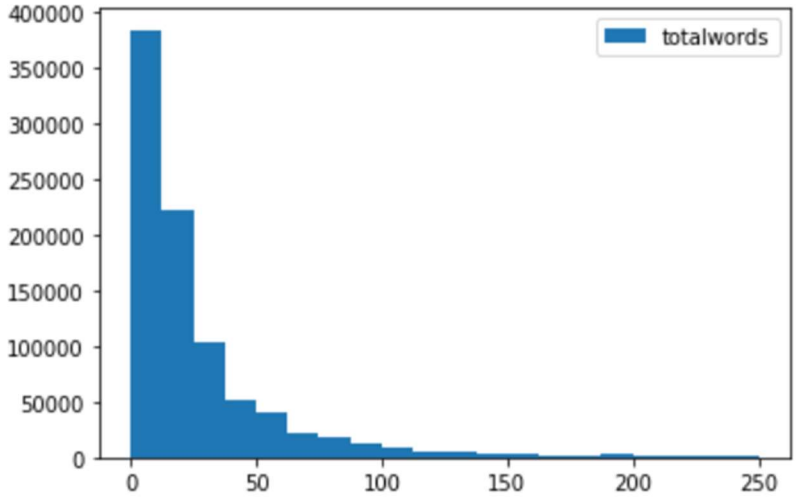


Fig. 1: Word count per comment skewed towards short in length

Contrary to the word count, the average word length seems to be following a normal distribution with 5 being the most common word length, as illustrated by **Figure 2**.

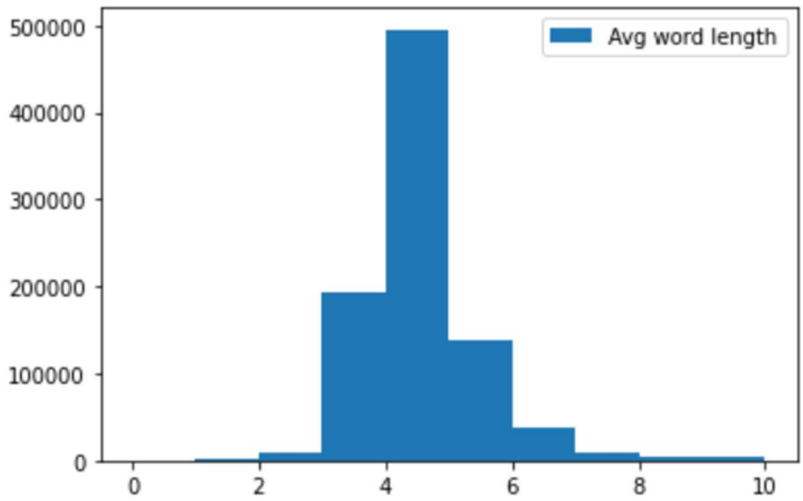


Fig. 2: Average word length is normally distributed

The immediate inference is that most authors use short words in their comments and after removing the stop words, an almost identical picture is being drawn in **Figure 3**.

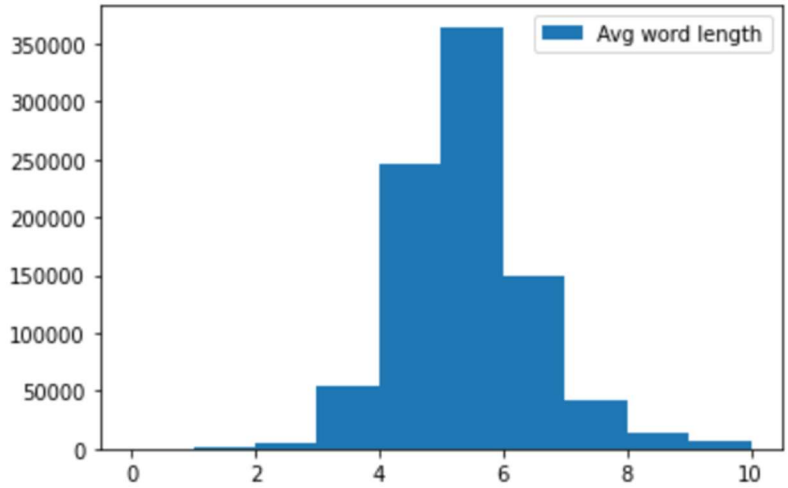


Fig. 3: Average word length after removing stop words

Analyzing the top 10 4-word combinations does not reveal much information other than the need to process our data carefully. Training our model on these combinations will likely lead to inaccurate predictions due to the fact that these phrases are expected to appear in all 40 topics. **Figure 4** shows the frequency of the most common 4-word combinations.

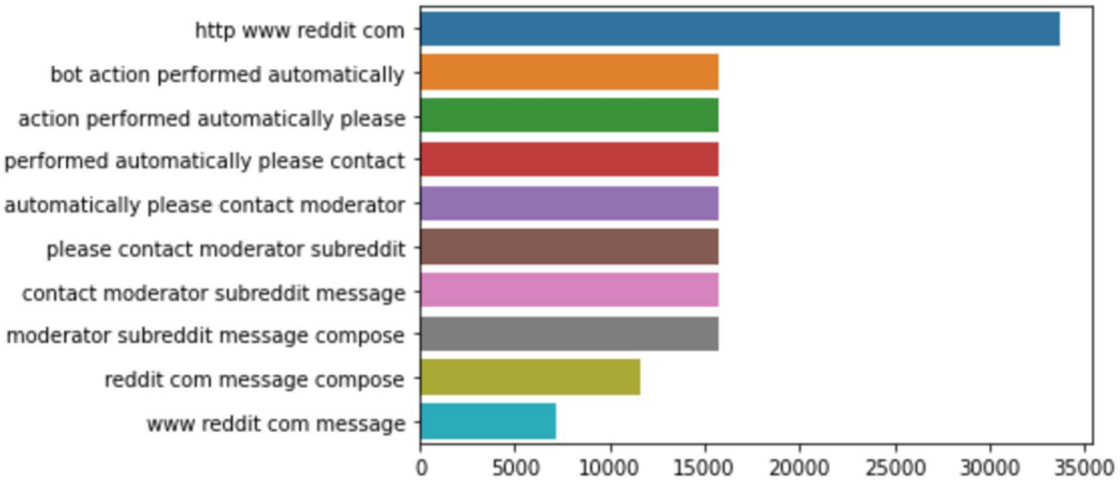


Fig. 4: Top 10 4-word combinations found in comments

In a next step, we can use the unsupervised learning method Latent Dirichlet Allocation (LDA) to model our topics and get a feeling for potential similarities between topics and the relative importance of the topic to the comments. In **Figure 5**, the area of the circle describes the importance and the distance between the circle centers represents the similarity between topics.

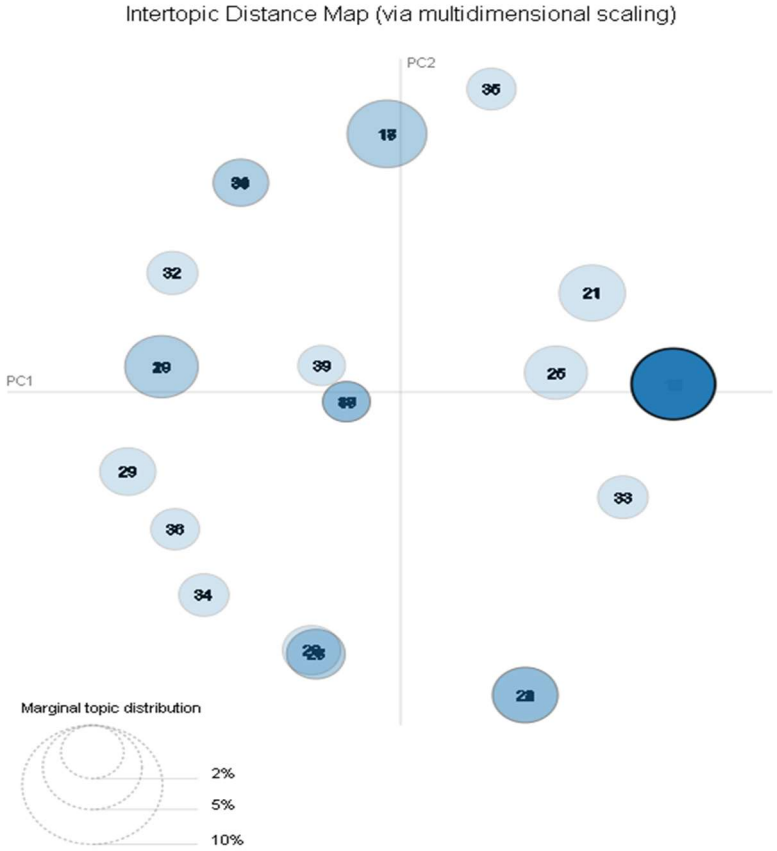


Fig. 5: Darker circles indicate (more) overlap between one or more topics

The above chart indicates that it may be challenging to find a definitive way to separate some topics. This may be due to the nature of the comments not yielding enough information that allows us to confidently classify them.

After the initial data exploration phase, our final approach to removing noise from the data includes the following steps:

- Remove non-alphabetic characters such as digits and punctuations
- Only keep words that are not found in NLTK's stop word collection
- Normalizing inputs by employing lemmatization to reduce each word to its base

Implementation

In our quest to find the model that can best predict which topic a comment belongs to, we considered a few machine as well as deep learning models. We hypothesized that either a Convolutional Neural Network (CNN) or Long Short-Term Memory (LSTM) model produces the best results. First, we tried a CNN with a network architecture as shown in **Figure 6**.

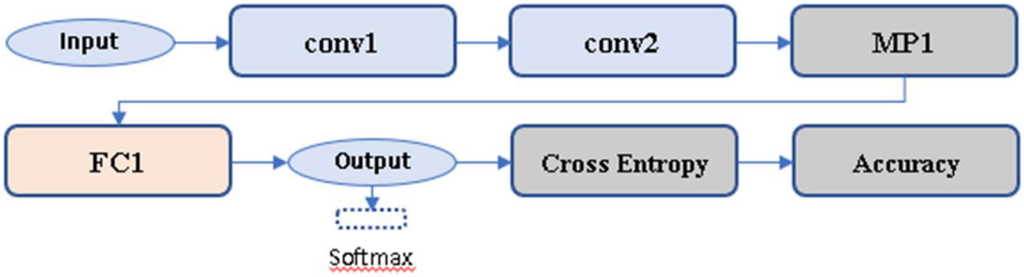


Fig. 6: CNN architecture

In the above computational graph, the acronyms can be read as follows:

- **Conv**: Convolutional layer.
- **FC**: Fully connected layer.
- **MP**: Max pooling layer.

After training our model, the validation accuracy of our model hovers around 5.3%, barely beating randomness (2.5%).

In a next attempt, we implemented a LSTM model, hoping to see an improvement in our prediction accuracy. The parameters we chose to initialize our model with are the following:

- The first hidden layer is an embedding layer with randomly assigned weights
- The next layer defines the dimension of the inner cells which is set to 100. A dropout rate is introduced during training which randomly ignores a number of layer outputs. This will result in a noisier training process that forces the nodes to compensate for and correct mistakes from prior layers, making the model more robust[3].

[3] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5357734/#bib4

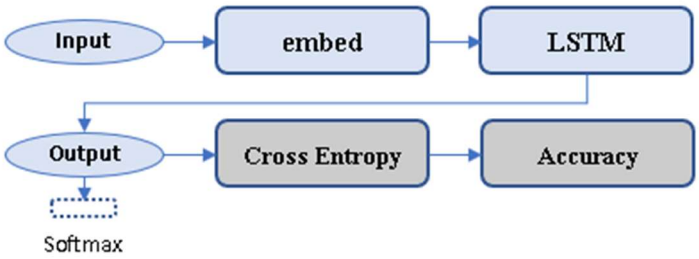


Fig. 7: LSTM network architecture

After implementing our LSTM model, our expectations of a better accuracy were not met. The model produces an accuracy of ~3.4%, performing worse than the CNN. Regardless of a too simple network structure, we wanted to move on and test other classifiers that could yield better results than the two deep learning models.

We turned to a Random Forest classifier and implemented it with these settings:

- We split the dataset into 5 equal folds, 4 of which the model will be trained on and 1 fold used to evaluate the model. Each fold will be used as test set once.
- To address the minor class imbalance in our target variable, we used the Synthetic Minority Over-sampling Technique (SMOTE) to level the playing field among all 40 classes.
- From python's "sklearn" package, we trained a Random Forest model on the train set, evaluated it with the test set, and recorded the resulting accuracy.
- To have a comparable figure across different models, the Cross Validation (CV) for all 5 models has been calculated.
- The best model's parameters are saved for easy access at a later stage

The Random Forest classifier was able to achieve a CV of ~35%. While this shows a significant improvement over the CNN and LSTM models, we wanted to try a pre-trained model and see if we could increase our accuracy even more.

In an effort to achieve the aforementioned target of increased accuracy, we chose to use the Bidirectional Encoder Representations from Transformers (BERT) model as a means of employing transfer learning. The model was initialized with the following settings:

- A train and test split of 80% and 20%, respectively
- Number of training epochs set to 3 and
- A static learning rate of 0.00001

At first, our model's runtime was exorbitantly high until we figured out a way to improve its performance by tweaking a few settings. This led to our ability to train the model on a greater number of train samples and simultaneously increasing its accuracy. To put it into numbers, we managed to go from a ~24% accuracy based on 10,000 train samples to a ~37% accuracy based on 100,000 train samples. Due to restricted access to computational resources, we were unable to train it further but it is to be expected that our model would have scaled up even more until an overfitting point would have been reached.

Conclusion

When working with textual data, it is imperative to clean the data thoroughly and to employ feature engineering techniques that best represent the textual information. It helps to analyze the complexity of your data and explore the composition visually through methods such as LDA (**Figure 5**). The preprocessed comments were fit to a Convolutional Neural Network, Long Short-Term Memory, Random Forest classifier, and a Bidirectional Encoder Representations from Transformers model to establish which model is best suited for predicting what topic they belong to. Through Cross Validation, we determined that BERT performs better than the other models, a result attributable to being a pre-trained model on 345 million parameters. Although it was only trained on 10,000 samples at the time of testing, BERT was the model we chose to enter the competition with. It was expected to score an accuracy of 24% and achieved an accuracy of 26.78%, slightly outperforming its expectations.

Chapter 2: Voice Tone Analysis

Project Statement

In this project we would access 6500 audio files. In a companion CSV file, for each file we see two responses: Res1 and Res2. Res1 takes 4 categorical values and Res2 takes 6 categorical values. The goal is to predict the classification of a voice. These two response variables characterize the voice in terms of the type of emotion, and the level of emotion. The data is unstructured, and the challenges are to clean and process it and extract important features from an audio file that can help us classify the tone.

Metrics

The metric used to evaluate our model is accuracy which considers each correct prediction with an equal weight.

$$Accuracy = \frac{Correct\ predictions}{Total\ predictions}$$

Techniques Considered

In preprocessing the data and building our model, we considered a number of different techniques, each having their own advantages to increase the performance and leading to better results. The following preprocessing techniques were explored:

- We employed Fourier Transformation as a means to preprocess the audio data and extract important features and characteristics.
- Librosa is a library in python that also extracts features.

To produce the best accuracy, we implemented several different machine learning and deep learning models. Each of the following has been tested on the same input:

- Random Forest (RF)
- Convolutional Neural Network (CNN)
- Feed Forward Neural Network.

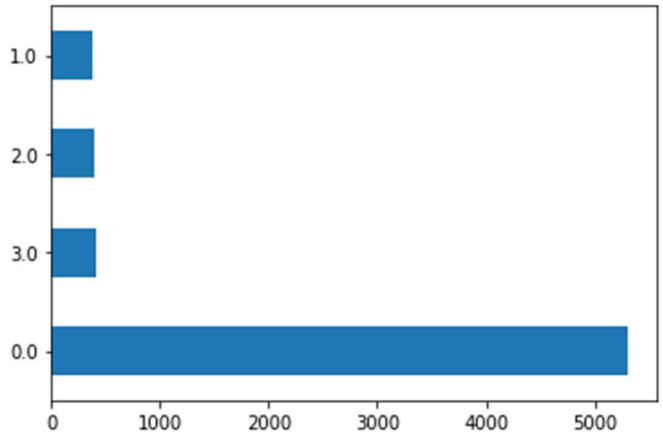
An ensemble of CNN and feed forward network was tested too but feed forward network produced the best results for us. All the networks were self-designed, and no pre-trained model was used. We do believe that highly granulated and better extraction could train a pre-trained model which we would have tested but were not able to due to time constraint.

We also would like to de-noise the samples before processing multiple techniques were tested but we could not achieve the desired result until we came across dolby denoise API which we might implement too.

Problem Analysis

Data Exploration

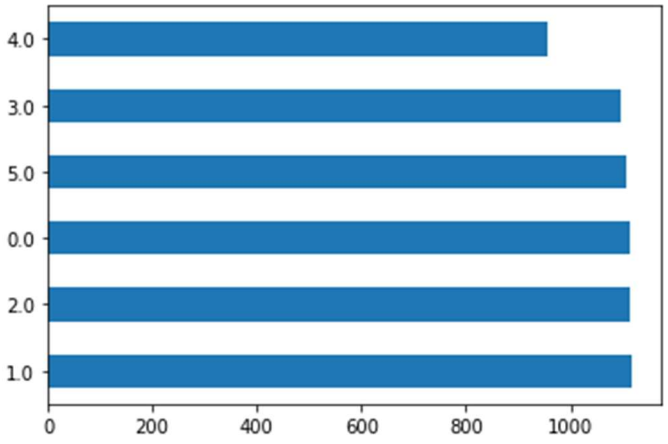
The dataset at hand consists of 6500 audio files. each belonging to a different class. We analyze the class distribution. First, let us take a look at Res1 data.



We see that the data for Res1 is quite imbalanced we will have to make decision on under sampling or not. We did not under sample due to two reasons.

- We will lose 80 percent of the data while under sampling, we only tested that for random forest.
- Neural networks performed better without under sampling as it learns from unbalanced data.

Next, we move on to Res2.



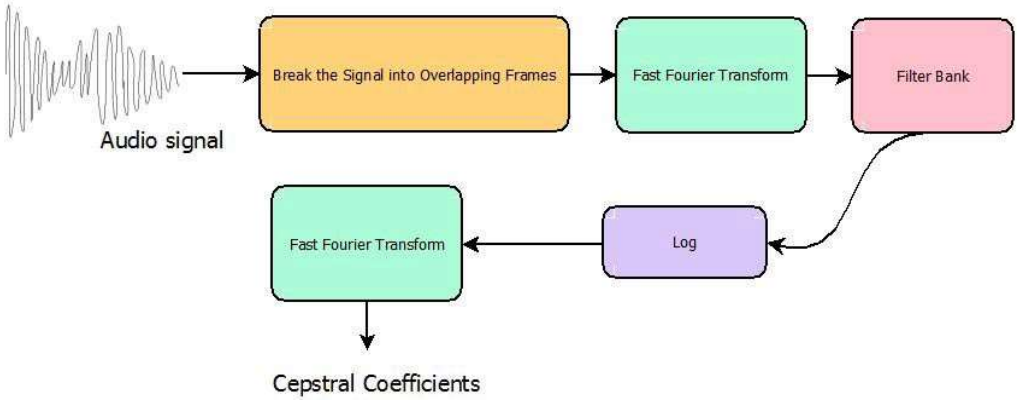
Here the data set is balanced, so no action is needed. We can move ahead to the modelling phase.

Implementation

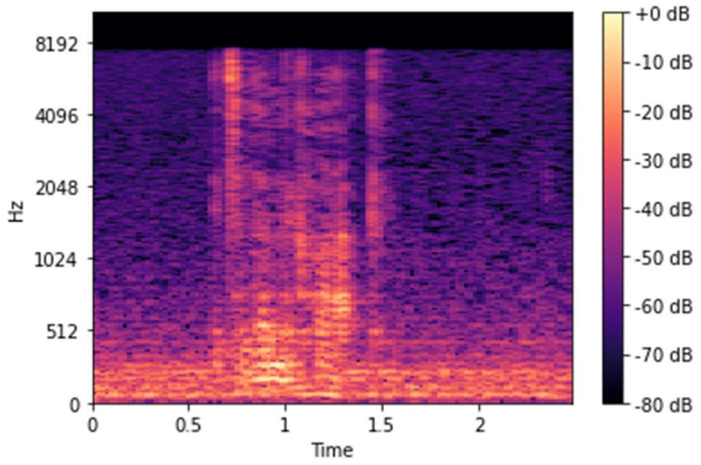
The most important aspect of this project is Fourier Transform. Any audio is first converted to frequency domain and then we can extract important features or characteristics of an audio using that. We tried to implement a model similar to Spotify's solving their cold start error.

Moving on to the implementation phase, we first discuss the features and what we considered for the purpose of our model.

- The most common and important feature extracted was **MFCC.** The envelope of the time power spectrum of the speech signal is representative of the vocal tract and MFCC accurately represents this envelope. A flowchart shows the way this is done:



- The next important feature is called **Chromagram.** It closely relates to the twelve different pitch classes. It helps in identifying the pitch of an audio and can assist in separating gender audio as well as certain emotions.
- Lastly, we look at the **Mel spectrogram.** The Mel spectrogram remaps the values in hertz to the Mel scale and is well suited for applications that need to model human hearing perception. Below is a plot of a Mel spectrogram from one of our samples, showing the time on the x-axis, the hertz on the y-axis and the color corresponding to the decibels (the brighter the color the higher the decibel). The Mel spectrogram was calculated for multiple granularities or bit rates, i.e., 512, 256 ad 128.



- We extracted multiple other features like Spectral centroid and contrast but those won't be discussed in detail as they weren't helpful in the classification.

After deciding the important features, we first tested Random Forest using MFCC as the input feature and Res1 and Res2 as output separately. The performance was poor for Res2 as shown by an accuracy of only 15%. These results are understandable as the data is unstructured and the features are incoherent.

The next step was to try a Feedforward Neural Network. Intuitively, this was the best solution that we had in mind after feature extraction was completed. Designing a neural network was done with a trial-and-error approach.

The final neural network structure contained four dense layers with inputs reducing in each layer and a dropout layer of varied values after each with ReLU as activation function for first three layers. Sigmoid was chosen as activation function for the last dense layer and the output layer uses SoftMax.
A selection on which feature to use for training was an important decision to make. The top features for training were MFCC and Mel spectrogram with different granularities. We finally decided to train the model on both and compare the results.

- For the Mel spectrogram with 128 granules and feedforward network, we achieved a validation accuracy of 45% and a test accuracy (with a holdout sample from the training set) of about 41% was obtained.
- For MFCC 120 granules for the Feedforward network, validation accuracy of 49% and test accuracy (with a holdout sample from the training set) of about 47% was obtained. Training on a bigger data set would yield better results.

Next, we decided to design and test a convolutional neural network using Mel spectrogram values as inputs. Our belief was higher granularity of Mel spectrogram should yield better results if we reshape our values and feed them as an image. It was realized that there were certain errors with this approach as the input data is not exactly an accurate representation of the image, even though the features take up a matrix value and corresponding values coincide for each input. Coming to the designing of CNN it depended on the granularity as well. Change in granularity did not affect the model to a great extent. We also trained it on 120 MFCC and results obtained were slightly poorer than the feedforward network.

- The best performance was for Mel spectrogram with 128 input and least was for 512.
- MFCC with 120 inputs also performed decent yielding a 43% accuracy for test data set on the hold out set and 42% on the test sample.

Lastly, we tried an approach to categorize and try to predict both the variable together because we believe it is only a correct classification if both the labels are predicted correctly. We make a column known as op which is just a string concatenation of Res1 and Res2. A test accuracy of 42% was achieved with 21 labels instead of 4 and 6 categorical variables.

We ran into an issue of not being able to categorize our output in respective classes as the answers were 16 and 19 which did not have labels. After receiving the test set results, we encoded the test using same methodology and 41% accuracy was achieved for the labels thus it's the true accuracy of our model.

Feedforward Network performed the best for us in all scenarios and MFCC 120 was the input that maximized the accuracy. MFCC with Chroma resulted in a similar accuracy thus we decided to stick to only MFCC as the input. Our expectation of Mel spectrogram were pretty high. We plan to convert each sound to image and then testing a pretrained model.

Our test accuracy during the first submission was poor due to an error in scaling of our test input and we later tried MFCC 120. These two changes uplifted our model accuracy from 14% to 50.2% for res2 and res1 accuracy was 82.5%

Conclusion

Our model gave pretty good results for Res1 and Res2 identification but more importantly the results with more and accurate labeling gave us good results too.

The problem was to classify the voice tone and We believe it is only a correct classification if both labels are predicted correctly that is the audio lies in its correct class in the classification. Our approach of string concatenation is not a perfect solution but inches us closer to a better classification for the give problem. The limitation of that approach is that there are only 21 combined labels. Thus, there are certain combinations that our model will never be trained on and can never guess such classifications.

Currently Feedforward network gave us best results but improving on certain aspects and building an ensemble technique and converting each sound into a better spectrogram can improve the model performance for this problem. We also believe that denoising the data will result in better feature extraction and Dolby denoise is great API available for this particular purpose. If the implementation of these steps is carried out we believe the accuracy can increase significantly.