Recognition System of Ethnic Groups from Speech – Ramallah, Palestine

Ahmad Hamad, Tarek Zidan, Mohammad Muwafi

[1] Department of Electrical and Computer Engineering, Birzeit University, Ramallah, Palestine
1180060@student.birzeit.edu, 1180404@student.birzeit.edu, 1180491@student.birzeit.edu

Abstract

In this report, we are going to demonstrate our spoken language processing course project at Birzeit University. After studying how sounds are generated in the human body, from the lungs through vocal tract until the lips, and how the sound is heard in the human ear, from the pinna, through outer, middle and until inner ear, we are required to demonstrate this process in the computer's domain. The project's main idea is to develop and evaluate a voice recognition system of ethnic groups from speech using voice and speaker recognition into two classes. This is possible when using speech signals properties, such as the envelope which carries important formants that have the speech identity, and which is generated by the vocal tract, these formants could be found at the low frequency components of speech spectrum to filter. While the spectrum of the speech frames also carries information and could be found at the higher frequency's components of speech spectrum, but it wouldn't be important in our application, since it indicates for the speaker's identity, and discriminate speakers from each other. This project was done using python programming language, where we used a large training data sent by the instructor to train our machine learning models, then another set of data was used in testing the model, and to evaluate its abilities.

Index Terms: voice recognition, voice and speaker, ethnic groups

1. Introduction

To develop a system that can identify the ethnic group of the British speaker living in Birmingham city as 'Asian' or 'White' English speaker we used machine learning models and trained them based on different properties such as Energy, Zero-crossing rate, Pitch frequency, and MFCCs. And of course, machine learning techniques.

Let's start with **basic parameters extraction**, these parameters could be easily calculated and would be very helpful in such applications. Parameters are estimated for speech frames between 10 to 20ms long, since speech signals are overlapped and seems to be non-stopping, so we divide them into frames, which we previously determine their period, then we follow up the entire process depending on the same window length.

1- Using energy property, we can discriminate between voiced and unvoiced components, where voiced components have lower energy, and unvoiced have higher energy. Note that energy of speech should be computed by dividing the speech into frames as we said earlier, this could be done by multiplying the signal with a suitable window function. Let's say we have N frames, then, we computer the sum of squared values of the signal samples in each frame. The following figure shows the process of calculating Energy:

$$E_m = \sum_n [s(n)\, w(m-n)]^2 \qquad w(n) = \begin{cases} 1 & 0 \le n \le N-1 \\ 0 & otherwise \end{cases}$$

$$\rightarrow \quad E_m = \sum_n [s(n)]^2\, h(m-n) \qquad h(n) = [w(n)]^2$$

2- Also, using zero crossing count rate is another property which is calculated for a block of N samples of speech. This property counts how many times the signal has crossed the time axis during a frame. Note that it is important to remove any DC component from the signal, so it is fixed on the x-axis itself and not shifted up or down so we can calculate the crossings correctly. A high zero crossing count indicates to a high frequency since the signal is alternating fast it will cross the axis more times. Also, High ZCC indicates to a voiced speech signal, while low ZCC indicates to an unvoiced speech signal. The following figure shows the process of calculating ZCC:

$$ZCC_i = \sum_{k=1}^{N-1} 0.5\,|\,sign(s[k]) - sign(s[k-1])\,|$$

3- A final basic parameter property is the pitch period, where pitch period is equal to the inverse of the fundamental frequency of vibration of the vocal cords, pitch is a property of voiced speech frames only, since unvoiced speech signals have zero pitch frequencies for not having vocal cords vibration. There are two ways to calculate pitch period, in time and frequency domains. Here, we will take care of time domain calculations only.

In time domain, we use one of two methods:

1- Short time autocorrelation function
2- Average magnitude difference function

In autocorrelation function, we measure how related are components of the same signal, by fixing one, and taking a copy if it, then sliding it on the fixed signal, using multiplication, we can see how components are related, note the when signals are exactly fixed on each other, same components will be multiplied, meaning that, a period has ended and a new period have started, this could be found on the highest multiplication component, and could be calculated as the following figure shows:

$$\phi(k) = \frac{1}{N} \sum_{n=0}^{N-1} s[n]s[n+k]$$

Another way, is average magnitude difference in time domain, where we subtract frequencies from each other, note that this operation is easier and faster than multiplication to perform on computer. Using

this property, we look for the lowest components to indicate new periods, since when signals are fixed on each other, the same components are subtracted which gives an answer close to zero depending on how accurate our calculations are. This could be calculated as the following figure shows:

$$D(k) = \frac{1}{N}\sum_{n=0}^{N-1} |s[n] - s[n+k]|$$

Now, in terms of **Machine Learning techniques,** we have used multiple supervised techniques which are:

1- **Logistic Regression (LR):** Logistic regression is one of the most popular supervised Machine Learning algorithms which is used for predicting the categorical dependent variable using a given set of independent variables. It predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

2- **Gaussian Naive Bayes (GNB):** Gaussian Naive Bayes is a variant of Naive Bayes Machine Learning technique that follows Gaussian normal distribution and supports continuous data. Note that Naive Bayes are a group of supervised machine learning classification algorithms based on the Bayes theorem which we have already taken in the course. It is a simple classification technique, but has high functionality. They find use when the dimensionality of the inputs is high. Complex classification problems can also be implemented by using Naive Bayes Classifier.

3- **Random Forest (RF):** Random Forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results in classification problems.

4- **Support Vector Machine (SVM):** Support Vector Machine is a supervised machine learning algorithm that can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is a number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well. Support Vectors are simply the coordinates of individual observation. The SVM classifier is a frontier that best segregates the two classes (hyper-plane/ line). Note that in our project we used SVC which is the linear version of SVM.

5- **K-Nearest Neighbor (KNN):** K-Nearest Neighbor is a supervised machine learning algorithm. The algorithm can be used to solve both classification and regression problem statements. The number of nearest neighbors to a new unknown variable that has to be predicted or classified is denoted by the symbol 'K'. We are often notified that you share many characteristics with your nearest peers, whether it be your thinking process, working etiquettes, philosophies, or other factors. As a result, we build friendships with people we deem similar to us.
The KNN algorithm employs the same principle. Its aim is to locate all of the closest neighbors around a new unknown data point in order to figure out what class it belongs to. It's a distance-based approach.

2. Background/Related Work

Voice recognition has evolved exponentially over the years. The purpose of voice recognition or sometimes called speaker identification, is to identify the person who is speaking. This can be done by extracting features of speech that differ between individuals due to physiology (shape and size of the mouth and throat) and also behavioral patterns (pitch, accent and style of speaking). Basic feature parameters in addition to number of Mel-Frequency Cepstrum Coefficients (MFCCs) most likely 13, are extracted from the sent datasets, and will be used in with the help of machine learning techniques that we mentioned earlier to evaluate a recognition system of ethnic groups from speech.
As a related work to our project, we read a few papers which are attached in the references section, these papers did projects that have the same idea as ours. We saw that some have used only Pitch frequency alone, without Energy and ZCC, others have used much larger amount of MFCCs than the 39 required MFCCs. Another interesting paper that we saw, a paper that talks about classifying speakers but for Arabic speakers, which gave us a motivation to try our project on different languages in the future. In general, what we have learned in this course, and what we have seen in the international papers is almost the same [7][8][9].

3. Methodology

The project was implemented in multi-stages as follows:

1) **Pre-processing stage**:
The .wav files was grouped into two classes; the first class was Asian sound and the other was White sound. The sampled files resembled at 8KHz. The silence at first and end of each voice was discarded. And then we normalized the files. Finally, we divide each .wav file into chunks.

2) **Feature extraction stage**:
In this stage, we extracted the 39 features from MFCC model using `librosa` library as we learned in the class such that we took 12 coefficients, their deltas, their double deltas, energy, delta of energy, and double delta of energy. And as extra features, we added the roll-off frequency since a lot of related works urge to use it. The roll-off frequency is defined for each frame as the center frequency for a spectrogram. However, after that, the data was transformed and becomes ready for the next stage.

3) **Classification and training stage**:

Before classification stage, the data split into training data and testing data in ratio of 80% - 20% as usual. Then the training data, was entered into different classifiers to build our models and use them for the evaluation stage.

We saw that it is better to use more than one classifier to compare their results and choose the best one of them. So, we used the following classifiers:

- Logistic Regression Classifier.
- Gaussian Naive Bayes Classifier.
- Random Forest Classifier.
- SVC Classifier (support vector machine).
- KNN Classifier (K-Neighbors).

4) **Evaluation stage**:
After building the models, we used the testing data to evaluate our results. The evaluation criteria were based on comparing Accuracy, Precision, Recall and F-measure of each model.

4. Experiments and Results

As we said previously, the data set was about .wav files for the ethnic group of the British speaker living in Birmingham city as 'Asian' or 'White' English speaker. We were going to recognize the .wav files into two classes using ML techniques. The features were extracted using MFCC model as explained above. The used classification models were 5 different models as explained previously. So, after building the model. We tested them using the same testing data.

As we said previously, we tested our data using 5 different models. And the following figures, show screenshots for our results:

```
========= The Confusion Matrix of LR ========
           True Positive  = 117
           False Positive = 61
           False Negative = 62
           True Negative  = 122
           Precision = 65.73
           Recall    = 65.363
           F1-score  = 65.546
           Accuracy  = 66.022
 ============================================
```

Figure 1 results of Logistic Regression Classifier

```
========= The Confusion Matrix of GNB ========
           True Positive  = 72
           False Positive = 106
           False Negative = 44
           True Negative  = 140
           Precision = 40.449
           Recall    = 62.069
           F1-score  = 48.98
           Accuracy  = 58.564
 ============================================
```

Figure 2 results of Gaussian Naive Bayes Classifier

```
========= The Confusion Matrix of RF ========
           True Positive  = 143
           False Positive = 35
           False Negative = 82
           True Negative  = 102
           Precision = 80.337
           Recall    = 63.556
           F1-score  = 70.968
           Accuracy  = 67.68
 ============================================
```

Figure 3 results of Random Forest Classifier

```
========= The Confusion Matrix of SVC ========
           True Positive  = 116
           False Positive = 62
           False Negative = 69
           True Negative  = 115
           Precision = 65.169
           Recall    = 62.703
           F1-score  = 63.912
           Accuracy  = 63.812
 ============================================
```

Figure 4 results of SVC Classifier

```
========= The Confusion Matrix of KNN ========
           True Positive  = 113
           False Positive = 65
           False Negative = 99
           True Negative  = 85
           Precision = 63.483
           Recall    = 53.302
           F1-score  = 57.949
           Accuracy  = 54.696
 ============================================
```

Figure 5 results of KNN Classifier

And to make the comparison more easier using eyes. See the following table to summarize the results.

Algorithm	Precision	Recall	Accuracy	F-measure
LR	65.73	65.363	66.022	65.546
GNB	40.449	62.069	58.564	48.98
RF	80.337	63.556	67.68	70.968
SVC	65.169	62.703	63.812	63.912
KNN	63.483	53.302	57.949	54.696

As we see in the above table, the results were not too bad. The Random Forest model gave us the best results. However, the models gave results that were a little bit close to RF model.

5. Conclusion

In this project, a speech recognition model to identify the ethnic group of the British speaker living in Birmingham city as 'Asian' or 'White' English speaker was implemented using MFCC model for feature extraction stage and 5 different classical machine learning classifiers for the classification stage. The MFCC was used to extract 12 coefficients with their delta and delta-delta in addition to the energy and their delta and delta-delta and the roll-frequency as explained above. The models were trained using these features and 80% of the data set and then tested using 20% of the remaining data.

The models were tested as shows in the above section. The Random Forest model gave the best results with P = 80.337%, R = 63.556%, Accuracy = 67.68% and F-measure = 70.968.

As a future plan, this project leads us to spend more time in learning deep learning models like CNN that can be used in the classification process to give a notice better performance than the classical models and trying to extract more features from the .wav files in order to get better performance and finally, we wish to deploy our models on a server with usable frontend website and strong server, and finally collecting a bigger dataset to generalize our result as much as we can and to avoid the overfitting or under fitting problems.

6. References

[1] https://www.javatpoint.com/logistic-regression-in-machine-learning#:~:text=Logistic%20regression%20is%20one%20of,of%20a%20categorical%20dependent%20variable.

[2] https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[3] https://iq.opengenus.org/gaussian-naive-bayes/

[4] https://www.analyticsvidhya.com/blog/2021/05/knn-the-distance-based-machine-learning-algorithm/#:~:text=The%20abbreviation%20KNN%20stands%20for,by%20the%20symbol%20'K'.

[5] https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/#:~:text=Random%20forest%20is%20a%20Supervised,average%20in%20case%20of%20regression.

[6] https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/#:~:text=What%20is%20the%20Support%20Vector,mostly%20used%20in%20classification%20problems.

[7] https://www.researchgate.net/publication/342603766_Speaker_ethnic_identification_for_continuous_speech_in_malay_language_using_pitch_and_MFCC

[8] https://core.ac.uk/download/pdf/86433096.pdf

[9] https://pubmed.ncbi.nlm.nih.gov/29201333/

7. Appendix

- The code was run on colab.

```
# impoort libraries.
import os
import sys
import librosa
import numpy as np
import pandas as pd
import librosa.display
import matplotlib.pyplot as plt
import pickle

from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.impute import SimpleImputer
from sklearn.svm import SVC
from sklearn import svm
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier

import tensorflow as tf

# split .wav file into chunks.
def split(list_a, chunk_size):
    for i in range(0, len(list_a), chunk_size):
        yield list_a[i:i + chunk_size]

# extract feautres and concatinate them.
def concatenate_features(S, sr):
    mfcc = librosa.feature.mfcc(S, n_mfcc=12, sr=sr).flatten()
    delta_mfcc = librosa.feature.delta(mfcc).flatten()
    delta2_mfcc = librosa.feature.delta(mfcc, order=2).flatten()
    energy = librosa.feature.rms(y=S).flatten()
    delta_energy = librosa.feature.delta(energy).flatten()
    delta2_energy = librosa.feature.delta(energy, order=2).flatten()
    roll_off = librosa.feature.spectral_rolloff(y=S,
sr=sr).flatten()

    feature_vector = np.concatenate((mfcc, delta_mfcc, delta2_mfcc,
energy, delta_energy, delta2_energy, roll_off,), axis=0)
    return list(feature_vector)


def get_feature_vector(file_path):
    vectors = []
    S, sr = librosa.load(file_path, sr=8000)

    # plt.figure(figsize=(12,4))
    # librosa.display.waveplot(S, sr=8000)

    S = librosa.effects.trim(S)[0]
    S = librosa.util.normalize(S)

    # plt.figure(figsize=(12,4))
    # librosa.display.waveplot(S, sr=8000)

    # sys.exit(0)

    temp = list(split(S, sr * 1 * 60))
    for vec in temp:
        vec = librosa.util.fix_length(vec, size=2 * sr * 60,
mode="edge")
        vectors.append(concatenate_features(vec, sr))
    return vectors

# store .wav files and extrat them.
```

```
def process_file(dir_path, type):
    res = []
    for path in os.listdir(dir_path):
        if os.path.isfile(os.path.join(dir_path, path)):
            res.append(dir_path + '/' + path)
    for file in res:
        vectors = get_feature_vector(file)
        for vec in vectors:
            data.append(vec)
            label.append(type % 2)



def prepare_data():
    process_file(r'/content/drive/MyDrive/Data set/Training/White',
0)
    process_file(r'/content/drive/MyDrive/Data set/Training/Asian',
1)
    process_file(r'/content/drive/MyDrive/Data set/Testing/White',
2)
    process_file(r'/content/drive/MyDrive/Data set/Testing/Asian',
3)

# print confusion matrix.
def print_confusion_matrix(y_test, spam_prediction, string):
    cm = confusion_matrix(y_test, spam_prediction)
    tp = cm[0][0]
    fp = cm[0][1]
    fn = cm[1][0]
    tn = cm[1][1]
    pr = tp / (tp + fp)
    rec = tp / (tp + fn)
    print("========= The Confusion Matrix of", string, "========")
    print("           True Positive  =", tp)
    print("           False Positive =", fp)
    print("           False Negative =", fn)
    print("           True Negative  =", tn)
    print("           Precision =", round(pr * 100, 3))
    print("           Recall    =", round(rec * 100, 3))
    print("           F1-score  =", round((2 * pr * rec) / (pr +
rec) * 100, 3))
    print("           Accuracy  =", round(accuracy_score(y_test,
spam_prediction) * 100, 3))
    print("=============================================")

prepare_data()

if __name__ == '__main__':

    data = np.array(data, dtype=object)
    label = np.array(label)

    data = data.astype('float32')
    label = label.astype('int')

    x_train, x_test, y_train, y_test = train_test_split(data, label,
test_size=0.2, random_state=0)

    classifier = LogisticRegression()
    classifier.fit(x_train, y_train)
    pickle.dump(classifier, open("LR_model.sav", 'wb'))
    sound_prediction_LR = classifier.predict(x_test)

    classifier = GaussianNB()
    classifier.fit(x_train, y_train)
    pickle.dump(classifier, open("GNB_model.sav", 'wb'))
    sound_prediction_GNB = classifier.predict(x_test)

    classifier = RandomForestClassifier(n_estimators=10,
criterion='entropy', random_state=0)
    classifier.fit(x_train, y_train)
    pickle.dump(classifier, open("RF_model.sav", 'wb'))
    sound_prediction_RF = classifier.predict(x_test)

    classifier = SVC(kernel='linear', random_state=0)
    classifier.fit(x_train, y_train)
    pickle.dump(classifier, open("SVC_model.sav", 'wb'))
    sound_prediction_SVC = classifier.predict(x_test)

    # classifier = svm.SVR()
    # classifier.fit(x_train, y_train)
    # sound_prediction_SVR = classifier.predict(x_test)

    classifier = KNeighborsClassifier(n_neighbors=2,
metric='minkowski', p=2)
    classifier.fit(x_train, y_train)
    pickle.dump(classifier, open("KNN_model.sav", 'wb'))
    sound_prediction_KNN = classifier.predict(x_test)

    print_confusion_matrix(y_test, sound_prediction_LR, "LR")
    print_confusion_matrix(y_test, sound_prediction_GNB, "GNB")
    print_confusion_matrix(y_test, sound_prediction_RF, "RF")
    print_confusion_matrix(y_test, sound_prediction_SVC, "SVC")
    print_confusion_matrix(y_test, sound_prediction_KNN, "KNN")

def load_model(file_name, x_test, y_test):
    loaded_model = pickle.load(open(file_name, 'rb'))
    sound_prediction = loaded_model.predict(x_test)
    print_confusion_matrix(y_test, sound_prediction, "##")

load_model('/content/GNB_model.sav', x_test, y_test)
```