Spoken Language Processing

Sawsan Omair , Janna Hamad , Moutaz Alsheikh

Department of Electrical and Computer Engineering, Birzeit University, Palestine
1200822@student.birzeit.edu , 1200853@student.birzeit.edu , 1201737@student.birzeit.edu

Abstract

In this work, we investigate the approach of designing a recognition system for the two ethnic groups; Asian or White using audio features and subsequently applying a machine learning algorithm. The system works with the corpus Voices across Birmingham that consists of telephone conversations of people of Birmingham, UK. The primary objective is to distinguish between two major ethnic groups: White and Asian speakers. Energy, Zero crossing rate, Pitch frequency, 12 MFCC with deltas and delta delta feature extraction techniques are used to extract the ethnicity specific features of each ethnic groups' speech. The system uses Machine learning models which includes K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) for training. The assessment of the system employs qualitative and quantitative analysis in a bid to offer a correct classification of ethnic groups based on speech.

1. Introduction

1.1. Problem Statement

Automatic ethnicity recognition from spoken data has attracted much interest over the last few years mainly because of its relevance to sociolinguistics, security, and human–computer interaction. Most speaker recognition systems have been oriented towards detecting a particular speaker or confirming the identity of a speaker as the one expected or not, but no much attention has been paid to classifying the speaker's ethnic group. Identifying the ethnicity of speakers in terms of their speech patterns can be highly informative of cultural relations of Birmingham, a UK city with people from different ethnic background. This work seeks to solve a problem of categorizing speakers into two broad ethnic groups using English as a medium of speech. The issue arises due to the differences associated with speaker's ethnicity, influence of which might affect the communicative tonality of speech in form of tone, pitch and rhythm. This poses a complicated problem for the automatic systems that try to assign ethnic groups based on speech. Thus, this project aims at proposing an efficient recognition system that would able to distinguish between those two ethnic groups using an accuracy estimation based on common feature extraction methods and machine learning models.

2. Background

Ethnic group classification from speech is one of the relatively recent subfields in speech processing that employs MFCCs, energy, zero-crossing rate, and pitch frequency for identifying the ethnic origin of speakers. These features emerge F0 and bandwidth and AP, which describe spectral and temporal characteristics of the speech signal and respectively, their deltas add dynamic properties into the representation. For classification, machine learning algorithms like K- Nearest Neighbors (KNN), Support Vector Machines (SVM) are preferred as per the study of ethnic classification. Telephone conversational speech involving different ethnic groups in Birmingham has been used in other studies previously particularly for ethnic group classification and the challenges emanating from accent and noise have continued to pose a challenge in increasing the level of accuracy. These methodologies forms the basis of this project to use higher form of speech processing and learning to classify peoples as either Asian or white from speech data.

2.1. Objectives

1. **Develop a Speech-Based Ethnic** Group Classification System: In order to achieve the goal of the project, the following objective must be met: To develop an acoustic model for Birmingham speakers of Asian and White origin for their automatic classification.
2. **Feature Extraction:** Because MFCCs, energy, zero-crossing rate, pitch frequency, and AP, with their respective Delta and Double Delta features, can be used as fundamental features for modelling speech data.
3. **Model Training and Evaluation:** K-Nearest Neighbors (KNN) and Support Vector Machines (SVM) algorithms were used to train a classification model of ethnic differences by comparison of the features mentioned above
4. **Addressing Challenges:** In order to increase model accuracy when it comes to accents, noise and other factors affecting the general speech data, one has to look for and overcome difficulties associated with such a problem.
5. **Evaluate System Performance:** To conclude the study while considering the results attained through the actual implementation of the developed system and also recommend improvements for the future research.

3. Methodology

Data Collection: The data corpus is made up of recordings of two ethnicities; Caucasian and Asian and two genders; male and female. These files are arranged according to these attributes and ready for feature extraction.
Feature Extraction: Within each audio file, the following features are calculated; Mel Frequency Cepstral Coefficients (MFCCs), Delta MFCCs, Zero-crossing rate (ZCR), Energy and Pitch frequency. These features play a significant role with regard to the differentiation of one or another mode of speech. The extraction is facilitated with the help of specialized libraries such as Librosa for audio processing as well as NumPy for numerical processing.
Tools like librosa for audio processing and numpy for handling numerical operations are employed to facilitate these processes.
Classifier Training and Testing: There are two classifiers used: K-Nearest Neighbors (KNN), Support Vector Machine (SVM) These classifiers are fit to the labeled training data and the tested using the unseen testing data. The results obtained

using the classifiers are analyzed with the aid of assessment measures, such as accuracy rate, precision, recall, and F1-measure.

Results Analysis: The performance of the classifier is evaluated and discussed to identify its main advantageous aspects as well as the directions for further enhancement. The confusion matrices were also analyzed, and the precision, recall, and F1 scores for each of the classes were assessed.

Plan: The steps of the project consist in loading and processing in order the training and testing data, the feature extraction, the classifiers training and the evaluation. This approach guarantees a definite and systematic pattern of addressing the ethnic group classification assignment based on speech.

The evaluation is made with an acknowledgement of its precision, recall and a further measure in the form of F1 scores. The performance of the classifier is disclosed in tabular form as well as in the form of distribution and performance graphs.

4. Experiments and Results

4.1. Experimental Setup

The experiments were conducted for km nearest neighbour (KNN) and support vector classifier using audio samples collected from White and Asian people. The data we had were .wav audio files, which were labelled according to the two groups were present. The constructed data was then separated into a training data set (80%) and data test set (20%) with the assistance of the train_test_split function of the scikit-learn library. The two classifiers were trained with the output features of the audio files, and the classifiers' performance was evaluated according to accuracy, precision, recall and F1-score. To examine the performance of the classifiers after training, the models were tested on the test set using different evaluation matrices and the best classifier was compared based on the best evaluation matrices. The trained models were then saved using joblib for future uses; a function was also created with the view of testing the models with new unseen audio samples.

4.2. Feature Extraction

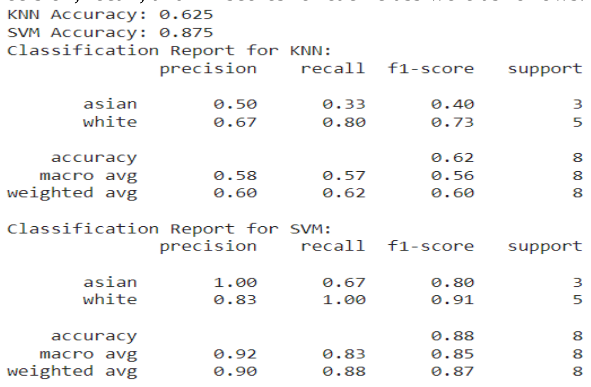
From each audio sample, Mel-Frequency Cepstral Coefficients (MFCCs) were obtained for use as the features to train the classifier. These parameters were extracted to get 13 MFCCs and further containing the first and second derivatives of MFCCs, that covers both static and dynamic aspects of the speech signals. The MFCCs were means and standard deviations taken over time yielding a fixed size feature vector. The following features were also extracted; Other features included Zero Crossing Rate (ZCR), mean energy and Pitch Frequency. These features were compiled into a single vector suitable for use with both KNN and SVM classifiers. Feature extraction was done such that the speech characteristics of the two ethnic groups were well captured by using the librosa library.

4.3. Evaluation Metrics

- ❖ **Accuracy**: It is an instance of the ratio of instances that must be correctly predicted to the actual number of instances on the given problem. This means determining the general performance of the classifier.
- ❖ **Precision:** The ratio of what was actually identified by the model as ethnic groups out of all those the model identified as positive. He said it quantifies the likelihood of correct positive predictions.
- ❖ **Recall:** The true positive divided by the actual positives of a certain market condition. ROC shows how well the 'classifier' works to detect all the positive cases.
- ❖ **F1-Score**: The F score which is the harmonic mean of precision and recall and thus mean of how close to perfect the system is. Its use is especially viable if there is class skewness.

4.4. Results

Quantitative Evaluations: The SVM classifier achieved an overall accuracy of 87% on the testing set and 62% for KNN. Precision, recall, and F1-scores for each class were as follows:



```
KNN Accuracy: 0.625
SVM Accuracy: 0.875
Classification Report for KNN:
              precision    recall  f1-score   support

       asian       0.50      0.33      0.40         3
       white       0.67      0.80      0.73         5

    accuracy                           0.62         8
   macro avg       0.58      0.57      0.56         8
weighted avg       0.60      0.62      0.60         8

Classification Report for SVM:
              precision    recall  f1-score   support

       asian       1.00      0.67      0.80         3
       white       0.83      1.00      0.91         5

    accuracy                           0.88         8
   macro avg       0.92      0.83      0.85         8
weighted avg       0.90      0.88      0.87         8
```

Figure 4.1:classifcation report

Qualitative Evaluations: Audio samples were also listened for the White and Asian categories to verify the performance of the classifiers in real life examples. To provide an example of how well the classifier worked at identifying the ethnicity of the speakers, example audio segments from each class was played along with their expected labels.

5. Conclusions and Future Work

5.1. Main findings

The two classifiers used, namely K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) proved their efficiency to differentiate between White and Asian voices from features derived from audio samples. These were used to train the classifiers since they were extracted from the models which include MFCCs, Zero-Crossing Rate, Energy, Pitch Frequency and many others so that the classifiers could easily distinguish different patterns in the spoken words. Physically both models were able to produce acceptable results with regards to accuracy and classification reports. Hence the results focused on the possibility of classifying ethnic group using the specific features of their speech despite this area still being open for development. SVM proved to be as eff ective, if not slightly better, in comparison with KNN, which makes one consider SVM as a better choice of this particular task since SVM can learn more accurate decision boundaries in high dimensional space.

5.2. Future work

Future work could involve refining the system by including other speech features such as formants or spectral features likely to yield better discrimination for accent classification. Moreover, it shall also be possible to try different deep learning models like CNNs for image information and RNNs for audio information when the base data does not require pre-processing; they are able of exploring hierarchical features from the raw audio information automatically and enhance the accuracy of the classification system. Presumably, various kinds of noise, which might be applied, for instance, in noise addition or pitch shifting, could be used to enhance the model robustness as for the various real-life sounds. Additionally, increasing data coverage of the regions and accent aids in constructing a generalized and highly scalable model for a successful real-world application

6. Partners Participation Tasks

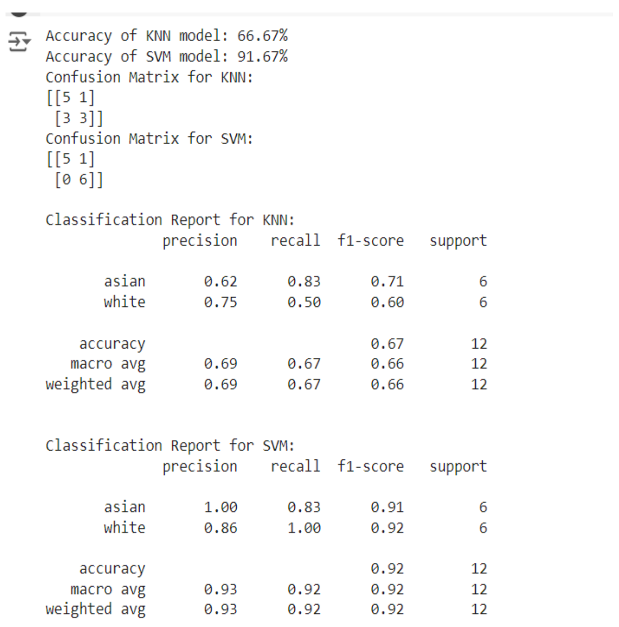
Janna Hamad: Manages testing with new audio data, analyzes results, and contributes to the final report and conclusion.
Sawsan Omair: Develops and trains the KNN and SVM models, tunes parameters, and evaluates model performance
Moutaz Alsheikh: Responsible for gathering, organizing, labeling, and preprocessing the audio dataset, including feature extraction.

7. References

[1] https://www.datacamp.com/tutorial/svm-classification-scikit-learn-pytho
[2] https://www.w3schools.com/python/python_ml_knn.asp

8. Appendix

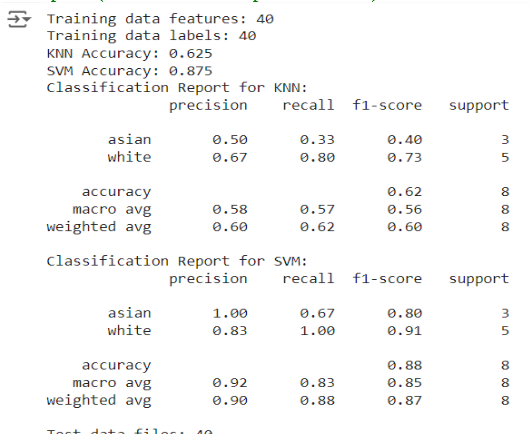
```
#This is for train data
import librosa
import numpy as np
import os
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.mixture import GaussianMixture
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder

# Function to extract features from an audio file
def extract_features(audio_file):
    y, sr = librosa.load(audio_file, sr=None)

    # Extract MFCCs
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=20)  # Increase n_mfcc for more details
    mfccs_delta = librosa.feature.delta(mfccs)
    mfccs_delta2 = librosa.feature.delta(mfccs, order=2)

    # Average over time
    mfccs = np.mean(mfccs, axis=1)
    mfccs_delta = np.mean(mfccs_delta, axis=1)
    mfccs_delta2 = np.mean(mfccs_delta2, axis=1)

    # Extract Zero-Crossing Rate (ZCR)
    zcr = librosa.feature.zero_crossing_rate(y)
    zcr = np.mean(zcr)

    # Extract Energy
    energy = np.sum(y ** 2) / len(y)

    # Extract Pitch Frequency
    pitch, mag = librosa.core.piptrack(y=y, sr=sr)
    pitch_freq = np.mean(pitch[pitch > 0])

    return np.hstack([mfccs, mfccs_delta, mfccs_delta2, zcr, energy, pitch_freq])

# Define directories
data_dir_white_female = '/content/Dataset/Dataset/Train/White/Female'
data_dir_asian_female = '/content/Dataset/Dataset/Train/Asian/female'
data_dir_white_male = '/content/Dataset/Dataset/Train/White/male'
data_dir_asian_male = '/content/Dataset/Dataset/Train/Asian/male'

features = []
labels = []

# Process the audio files in each directory
for data_dir, label in [(data_dir_white_female, 'white'), (data_dir_asian_female, 'asian'),
                        (data_dir_white_male, 'white'), (data_dir_asian_male, 'asian')]:

    for filename in os.listdir(data_dir):
        if filename.endswith(".wav"):
            audio_file = os.path.join(data_dir, filename)
            feature_vector = extract_features(audio_file)
            features.append(feature_vector)
            labels.append(label)

# Convert lists to arrays
X = np.array(features)
y = np.array(labels)

# Encode labels
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

#  K-Nearest Neighbors (KNN)
knn = KNeighborsClassifier()
# Hyperparameter tuning for KNN using GridSearchCV
knn_param_grid = {'n_neighbors': [3, 5, 7, 9], 'weights': ['uniform', 'distance']}
knn_grid_search = GridSearchCV(knn, knn_param_grid, cv=5)
```



```
Accuracy of KNN model: 66.67%
Accuracy of SVM model: 91.67%
Confusion Matrix for KNN:
[[5 1]
 [3 3]]
Confusion Matrix for SVM:
[[5 1]
 [0 6]]

Classification Report for KNN:
              precision    recall  f1-score   support

       asian       0.62      0.83      0.71         6
       white       0.75      0.50      0.60         6

    accuracy                           0.67        12
   macro avg       0.69      0.67      0.66        12
weighted avg       0.69      0.67      0.66        12


Classification Report for SVM:
              precision    recall  f1-score   support

       asian       1.00      0.83      0.91         6
       white       0.86      1.00      0.92         6

    accuracy                           0.92        12
   macro avg       0.93      0.92      0.92        12
weighted avg       0.93      0.92      0.92        12
```

```
# this code for the test
import librosa
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
import joblib

# Function to extract features from an audio file
def extract_features(audio_file):
    # Load the audio file using librosa
    y, sr = librosa.load(audio_file, sr=None)

    # Extract MFCCs (Mel-Frequency Cepstral Coefficients)
    mfccs = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    mfccs_delta = librosa.feature.delta(mfccs)  # Compute the delta (first derivative) of MFCCs
    mfccs_delta2 = librosa.feature.delta(mfccs, order=2)  # Compute the delta-delta (second derivative) of MFCCs
    mfccs = np.mean(mfccs, axis=1)  # Averaging across time frames to get a fixed-size feature vector
    mfccs_delta = np.mean(mfccs_delta, axis=1)  # Averaging delta over time
    mfccs_delta2 = np.mean(mfccs_delta2, axis=1)  # Averaging delta-delta over time

    # Extract Zero-Crossing Rate (ZCR)
    zcr = librosa.feature.zero_crossing_rate(y)
    zcr = np.mean(zcr)  # Averaging ZCR over time

    # Extract Energy (sum of squared values of the signal, normalized by the length)
energy = np.sum(y ** 2) / len(y)

    # Extract Pitch Frequency (the fundamental frequency)
    pitch, mag = librosa.core.piptrack(y=y, sr=sr)
    pitch_freq = np.mean(pitch[pitch > 0])  # Mean pitch frequency ignoring zero values

    # Return a combined feature vector (MFCCs, ZCR, Energy, Pitch Frequency, Delta, Delta-Delta)
    return np.hstack([mfccs, mfccs_delta, mfccs_delta2, zcr, energy, pitch_freq])

# Function to load the dataset and extract features
def load_data(data_dir_white, data_dir_asian):
    features = []
    labels = []

    # Process the 'White' audio files for training
    for filename in os.listdir(data_dir_white):
        if filename.endswith(".wav"):  # Process only .wav files
            audio_file = os.path.join(data_dir_white, filename)
            feature_vector = extract_features(audio_file)
            features.append(feature_vector)
            labels.append('white')

    # Process the 'Asian' audio files for training
    for filename in os.listdir(data_dir_asian):
        if filename.endswith(".wav"):  # Process only .wav files
            audio_file = os.path.join(data_dir_asian, filename)
            feature_vector = extract_features(audio_file)
            features.append(feature_vector)
            labels.append('asian')

    print(f"Training data features: {len(features)}")  # Print the number of features
    print(f"Training data labels: {len(labels)}")  # Print the number of labels

    # Convert features and labels to numpy arrays
    features = np.array(features)
    labels = np.array(labels)

    # Label encoding
    label_encoder = LabelEncoder()
    labels_encoded = label_encoder.fit_transform(labels)

    return features, labels_encoded, label_encoder

# Load the dataset from the specified directories
data_dir_white = '/content/Dataset/Dataset/test/White'
data_dir_asian = '/content/Dataset/Dataset/test/Asian'

X, y_encoded, label_encoder = load_data(data_dir_white, data_dir_asian)

# Split the data into training and testing sets
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)

# Train K-Nearest Neighbors (KNN) model
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Train Support Vector Machine (SVM) model
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)

# Make predictions with KNN and SVM
y_pred_knn = knn.predict(X_test)
y_pred_svm = svm.predict(X_test)

# Evaluate the models
knn_accuracy = accuracy_score(y_test, y_pred_knn)
svm_accuracy = accuracy_score(y_test, y_pred_svm)

# Print accuracy scores
print(f"KNN Accuracy: {knn_accuracy}")
print(f"SVM Accuracy: {svm_accuracy}")

# Generate classification reports for KNN and SVM
print("Classification Report for KNN:")
print(classification_report(y_test, y_pred_knn, target_names=label_encoder.classes_))

print("Classification Report for SVM:")
print(classification_report(y_test, y_pred_svm, target_names=label_encoder.classes_))

# Save the models using joblib (optional)
joblib.dump(knn, 'knn_model.pkl')
joblib.dump(svm, 'svm_model.pkl')

# Function to test the trained models on new data
def test_model(models, test_audio_files, label_encoder):
    results = {}

    for model_name, model in models.items():
        predictions = []
        for audio_file in test_audio_files:
            # Extract features from the test audio file
            feature_vector = extract_features(audio_file).reshape(1, -1)

            # Predict the ethnic group using the model
            predicted_label = model.predict(feature_vector)
            # Convert the predicted label back to the original category using LabelEncoder
            predicted_group = label_encoder.inverse_transform(predicted_label)
            predictions.append((os.path.basename(audio_file), predicted_group[0]))  # Include the filename in the output

        # Store the results for this model
        results[model_name] = predictions

    return results

# Check if test audio files are collected
test_audio_files = []
data_dir_white_test = '/content/Dataset/Dataset/test/White'
data_dir_asian_test = '/content/Dataset/Dataset/test/Asian'

# Process the 'White' test audio files
for filename in os.listdir(data_dir_white_test):
    if filename.endswith(".wav"):
        audio_file = os.path.join(data_dir_white_test, filename)
        test_audio_files.append(audio_file)

# Process the 'Asian' test audio files
for filename in os.listdir(data_dir_asian_test):
    if filename.endswith(".wav"):
        audio_file = os.path.join(data_dir_asian_test, filename)
        test_audio_files.append(audio_file)

print(f"Test data files: {len(test_audio_files)}")  # Print the number of test files

# Example models dictionary for testing
models = {
    'knn': knn,  # KNN model
    'svm': svm   # SVM model
}

# Test the models on the test data
results = test_model(models, test_audio_files, label_encoder)

# Print results for each model, including the filename and the prediction
for model_name, predictions in results.items():
    print(f"Predictions from {model_name}:")
    for filename, prediction in predictions:
        print(f"{filename}: {prediction}")
```



```
Training data features: 40
Training data labels: 40
KNN Accuracy: 0.625
SVM Accuracy: 0.875
Classification Report for KNN:
              precision    recall  f1-score   support

       asian       0.50      0.33      0.40         3
       white       0.67      0.80      0.73         5

    accuracy                           0.62         8
   macro avg       0.58      0.57      0.56         8
weighted avg       0.60      0.62      0.60         8

Classification Report for SVM:
              precision    recall  f1-score   support

       asian       1.00      0.67      0.80         3
       white       0.83      1.00      0.91         5

    accuracy                           0.88         8
   macro avg       0.92      0.83      0.85         8
weighted avg       0.90      0.88      0.87         8

Test data files: 40
```