Simple Acoustic Palestinian Regional Accent Recognition System

Abdallah Mohammad, Mahmoud Atia, Ola Salem

Electrical and Computer Engineering Dept. Birzeit University
Mahmoudatia024@gmail.com 1192519. abdallahdaoud37@gmail.com 1190515

olasalem304@gmail.com 1192439

(Ola Work)

Abstract

This work investigates the use of machine learning methods on audio characteristics for the classification of spoken accents. Four distinct accents are represented in the dataset: Ramallah Reef, Hebron, Jerusalem, and Nablus. Preprocessing is done on the audio data to extract features, such as MFCCs (mel-frequency cepstral coefficients). The Support Vector Machine (SVM) machine learning model is trained and evaluated using the retrieved features. The SVM model was 70% accurate, however it had trouble differentiating between Ramallah Reef dialects and other accents. In order to solve this, a Grid search was investigated as a way to improve SVM's efficiency. The SVM demonstrated potential for enhancing the categorization of Ramallah Reef accents, successfully classifying the majority of accents with an accuracy of 75%.

1. Introduction

An essential component of automatic speech recognition systems is accent recognition, which helps with personalised applications and offers insights into linguistic variation. Palestinian accents are difficult to understand because of their slight variances in phonetic characteristics. The goal of this project is to create a system that can categorise speech segments into four distinct regional accents: Ramallah, Hebron, Jerusalem, and Nablus. The method uses MFCC to extract acoustic characteristics, then grid search is used in an SVM classifier to achieve the best possible parameter tuning.

2. Background

Accent recognition has garnered significant attention in the field of speech processing. Previous studies have explored various feature extraction techniques, such as MFCC, Linear Predictive Coding (LPC), and their combinations, to capture the phonetic characteristics of different accents. Classification methods range from traditional machine learning algorithms, including SVM and k-Nearest Neighbors (k-NN), to advanced deep learning models like Convolutional Neural Networks (CNNs). This project builds upon these foundations, focusing on MFCC and SVM due to their proven effectiveness in similar tasks.

2.1. Mel-Frequency Cepstral Coefficients (MFCC)

A common feature in speaker detection and automated speech is MFCCs. They come from a kind of sound processing known as cepstral analysis, which is calculating the logarithm of the estimated signal spectrum and applying the Fourier transform to it.

This is how the procedure is broken down:

- Pre-emphasis: The first step is to apply a pre-emphasis filter to the audio signal. This is usually a simple high-pass filter that emphasizes higher frequencies.
- Frame Blocking: The audio signal is divided into short frames, typically 20-40 milliseconds long. These frames are often overlapped to ensure a smoother transition between frames.
- Windowing: Each frame is multiplied by a window function, such as the Hamming window, to reduce spectral leakage.
- Fast Fourier Transform (FFT): The FFT is applied to each framed signal to convert it from the time domain to the frequency domain.
- Mel Filtering: The spectrum is then passed through a bank of filters spaced evenly on the mel scale, which is a perceptual scale of pitches. This process is meant to mimic the human ear's response to different frequencies.
- Log Compression: The log of the powers at each filter output is taken to mimic the human auditory system's response to loudness.
- Discrete Cosine Transform (DCT): Finally, a discrete cosine transform is applied to the log filter bank energies to decorrelate the features and obtain the cepstral coefficients.

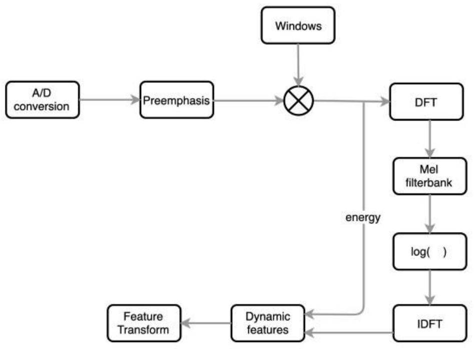

Figure 1 MFCC Technique for Speech Recognition

2.2. Support Vector Machine (SVM)

A strong and adaptable machine learning model for both classification and regression applications is the Support Vector Machine (SVM). Its capacity to identify the ideal hyperplane for effectively dividing various classes within the input space is its main strength.

SVM's functionality for classification tasks:

- Linear Separability: Support Vector Machines (SVM) perform best when the data can be split into two classes using a straight line in two dimensions, a plane in three dimensions, or a hyperplane in higher dimensions.
- Maximising Margin: Support Vector machines (SVM) seek to locate the hyperplane that maximises the margin, defined as the separation between the hyperplane and the closest data point from each class. The model's capacity for generalisation is strengthened by this margin maximization.
- Support Vectors: Support vectors are the data points that are closest to the hyperplane. These locations are utilised to maximise the margin and are essential for defining the hyperplane.
- Kernel Trick: By mapping the input space into a higher-dimensional space where the data may be linearly separable, a kernel function allows SVM to handle non-linearly separable data efficiently. Sigmoid, polynomial, radial basis function (RBF), and linear kernels are examples of common kernel functions.
- Regularisation: To balance the margin maximisation and the classification error, SVM has a regularisation parameter (C). Whereas a lower C value permits a bigger margin but more classification errors, a higher C value permits a smaller margin but fewer classification errors.
- Decision Function: By analysing the decision function, the SVM model may be trained to predict the class of incoming data points. The class with the highest decision function value receives the data point.

Because of their efficiency and adaptability, support vector machines (SVMs) are extensively employed in a wide range of domains, including bioinformatics, text categorization, and image classification. But they can be computationally demanding, particularly when dealing with big datasets, and careful hyperparameter selection is necessary to achieve the best results.

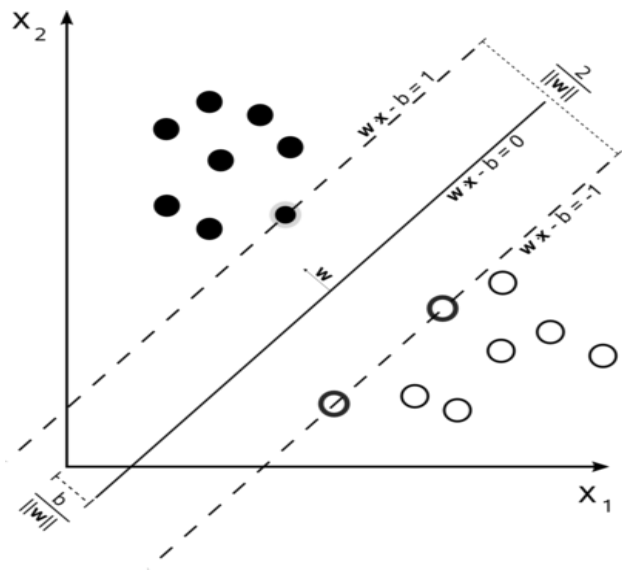

Figure 2 Support Vector Machines (SVM)

2.3. Grid search

One technique for fine-tuning hyperparameters in machine learning models, like Support Vector Machines (SVM), is grid search. Hyperparameters are settings made before training that have an impact on learning but are not directly learned during the training process.

C and gamma are two crucial hyperparameters in the SVM instance.

- The regularisation parameter, C, regulates the trade-off between minimising the decision boundary's complexity and obtaining a low error on the training set.
- Gamma is a non-linear hyperplane parameter. It may attempt overfitting if the gamma value is increased since it will attempt to match the training data set more precisely.

Grid search operates by first creating a grid of possible hyperparameter values, and then using cross-validation to assess the model's performance for each set of hyperparameters. The term "grid search" refers to the methodical way in which it searches over the grid to find the optimal set of hyperparameters based on a scoring criterion, like accuracy or F1 score.

(Abdallah Work)

3. Methodology

The system is designed with the following key components:

Our goal in this project is to create a system for recognizing regional accents in Palestinian speech using acoustic cues that are taken from speech. Mel-frequency cepstral coefficients (MFCCs), chroma features, and spectral contrast features are the three primary feature types used by the system.

Mel-frequency cepstral coefficients (MFCCs):

MFCCs are commonly employed in speech and audio processing to record the audio signals' spectrum properties.
From each audio segment, we use the librosa.feature.mfcc function to extract 80 MFCCs.
To generate a compact representation of the audio segment, the mean of these coefficients across the frames is calculated.

Chroma features:

The energy distribution of an audio segment's twelve pitch classes is represented by chroma characteristics.
Using the librosa.feature.chroma_stft function, we extract Chroma features and calculate the average over the frames.

Spectral Contrast features:

The amplitude difference between peaks and troughs in the spectrum of an audio stream is captured by spectral contrast characteristics.
Using the librosa.feature.spectral_contrast function, we extract Spectral Contrast features and calculate the mean over the frames.

Feature Combination:

For every audio segment, a single feature vector is created by concatenating the retrieved MFCCs, Chroma features, and Spectral Contrast features.

Training Models:

Radial basis function (RBF) kernel-equipped Support Vector Machine (SVM) classifier is what we employ.
The combined feature vectors from the training dataset are used to train the SVM model.

Evaluation:

A different testing dataset is used to assess the trained SVM model's ability to distinguish regional accents unique to Palestine.
The performance parameter we choose is accuracy, which quantifies the proportion of correctly identified occurrences.

The MFCCs are computed as follows:

- Pre-emphasis:

$$y(t) = x(t) - \alpha x(t-1)$$

- Framing and Windowing:

$$x_w(n) = x(n) \cdot w(n)$$

- Fast Fourier Transform (FFT) and Power Spectrum:

$$|X(k)|^2 = |\sum_{n=0}^{N-1} x_w(n) e^{-\frac{j2\pi kn}{N}}|^2$$

- Mel-filterbank:

$$H\_m\,(f) = \begin{cases} 0 & if\ f < f_{m-1} \\ \dfrac{f - f_{m-1}}{f_m - f_{m-1}} & if\ f_{m-1} \le f < f_m \\ \dfrac{f_{m+1} - f}{f_{m+1} - f_m} & if\ f_m \le f < f_{m+1} \\ 0 & if\ f \ge f_{m+1} \end{cases}$$

- Discrete Cosine Transform (DCT):

$$MFCC(n) = \sum_{K=0}^{K-1} \log \log \left(X(k)\right) \cos \left(\frac{\pi n(k+0.5)}{K}\right)$$

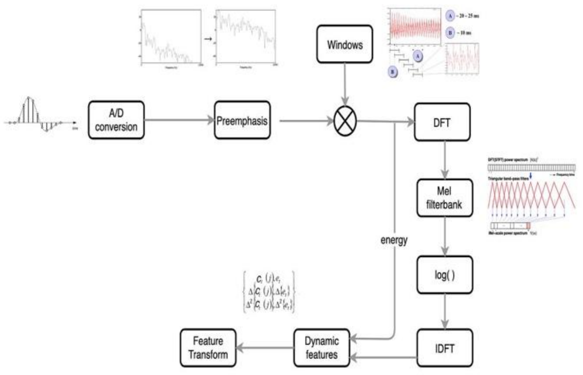

Figure 3 Speech Recognition

(Mahmoud Work)

4. Experiments and Results

4.1. Training Process

This system's initial step is to compute the feature vector that will be applied to the classification. Mel-Frequency Cepstral Coefficients (MFCCs) are the feature vector that we have chosen. We specifically took out twenty MFCC coefficients. The MFCC characteristics were extracted from each audio file by processing, and the mean value of these coefficients was computed for every file. For every audio sample, this produced a feature vector of 20 dimensions.

Because of its dependability in classification tasks, a Support Vector Machine (SVM) classifier was selected. We used a grid search to fine-tune the hyperparameters, specifically the regularisation parameter C, the kernel type, and the class weights, in order to maximise the performance of the SVM. The grid search looked at class weights ['balanced'], kernel types ['linear', 'rbf'], and C values of [0.1, 1, 10].

4.2. Testing Process

After training the SVM model, we used the testing dataset to assess the system. The trained SVM model was utilised to predict the accent class for every test file after extracting the MFCC features. The F1-score, accuracy, precision, and recall metrics were used to evaluate the classification performance. A confusion matrix was also created to offer a comprehensive picture of the classification outcomes.

4.3. Dataset

Speech samples from four Palestinian regions were included in the dataset: Ramallah-Reef, Hebron, Jerusalem, and Nablus. The quantity of.wav files submitted by each location was considerable. Training and testing sets of the pre-split data were loaded independently from designated directories.

4.4. Results

The test dataset was used to assess the system's performance, and the following tables and figures provide a summary of the findings.

4.4.1. Classification Metrics

Class	Precision	Recall	F1-Score	Support
Hebron	1.00	0.80	0.89	5
Jerusalem	1.00	1.00	1.00	5
Nablus	0.67	0.40	0.50	5
Ramallah-Reef	0.50	0.80	0.62	5
Accuracy			0.75	20
Macro Avg	0.79	0.75	0.75	20
Weighted Avg	0.79	0.75	0.75	20

Figure 4 Classification Metrics

4.5. Confusion Matrix

The confusion matrix provides a detailed breakdown of the classification results:

	Predicted Hebron	**Predicted Jerusalem**	**Predicted Nablus**	**Predicted Ramallah-Reef**
Hebron	4	0	0	1
Jerusalem	0	5	0	0
Nablus	0	0	2	3
Ramallah-Reef	0	0	1	4

Figure 5 Confusion Matrix

4.6. Discussion

Overall, the SVM classifier performed well, obtaining an accuracy of 75% on the test dataset. The efficacy of the classifier in differentiating between the accents of Hebron, Jerusalem, and Nablus was demonstrated by the exceptionally high precision, recall, and F1-score for these regions. With an F1-score of 0.50, Ramallah-Reef's performance was, nevertheless, worse. This disparity implies that either the dataset size for the Ramallah-Reef accent needs to be expanded to enhance model training, or the features employed could not be effectively capturing the distinctive qualities of this dialect.

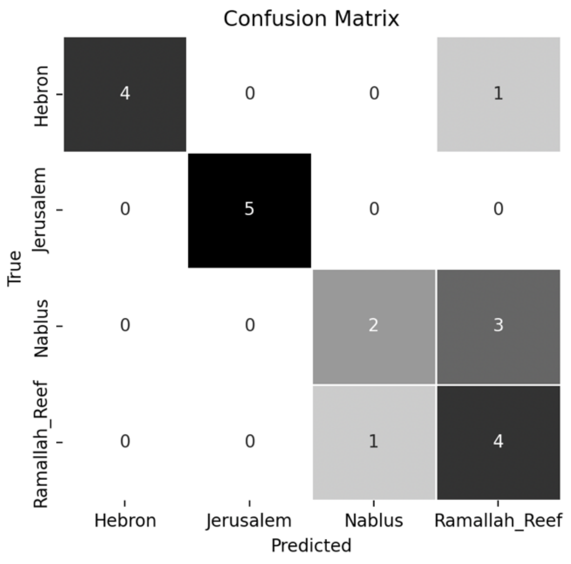

Figure 6 Confusion Matrix

4.7. Evaluation Metrics

Precision: 0.79%
Recall: 75.0%
F1-Score: 75.0%
Accuracy: 75.0%

These findings show that although the system works consistently well, there is still space for development, particularly with regard to the Ramallah-Reef accent.

5. Conclusion and Future Work

Using MFCC features and an SVM classifier, the developed acoustic accent recognition system achieves an 75% accuracy rate in distinguishing four regional accents from Palestine. Enhancing the feature extraction procedure, investigating alternative classification algorithms, and growing the dataset size for more reliable training could be the main goals of future research. Furthermore, adding more sophisticated methods like deep learning might help recognition performance even further.

Our goal in tackling these aspects is to develop a more precise and dependable system that can differentiate between the minute differences in regional dialects spoken in Palestine.

6. References

[1] Cortes, Corinna; Vapnik, Vladimir (1995). "Support-vector networks"
[2] GeeksForGeeks : SVM Hyperparameter Tuning using GridSearchCV | ML
[3] Analytics Vidhya : MFCC Technique for Speech Recognition

(Group Work)

7. Appendix

I upload The Code on GitHub and This is The Link:

- https://github.com/mahmoud024/Spoken-Project/blob/main/Main.py

```
#------------------------ Library -------------------------
---------#
import os
import glob
import librosa
import numpy as np
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score,
confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt
import pickle
from concurrent.futures import ThreadPoolExecutor


#--------------------------------------------------------#
#---------------------- استخراج الميزات ----------------------#


# Function to extract features from a single audio file
def extract_features(file_path):
    try:
        y, sr = librosa.load(file_path, sr=None)
        # Extract MFCCs = 70%
        mfccs = librosa.feature.mfcc(y=y, sr=sr,
n_mfcc=80)
        mfccs_mean = np.mean(mfccs, axis=1)
        # Extract Chroma features = 70%
        chroma = librosa.feature.chroma_stft(y=y, sr=sr)
        chroma_mean = np.mean(chroma, axis=1)
        # Extract Spectral Contrast = 65%
        contrast = librosa.feature.spectral_contrast(y=y,
sr=sr)
        contrast_mean = np.mean(contrast, axis=1)
        # Combine all features
        combined_features = np.concatenate((mfccs_mean,
chroma_mean, contrast_mean))
        return combined_features
    except Exception as e:
        print(f"Error loading {file_path}: {e}")
        return None
```

```
#--------------------------------------------------------#
#----------------- تحميل ملفات الصوت--------------------------#

# Function to load audio files and extract acoustic
features
def load_audio_files(base_path):
    features = []
    labels = []
    accents = ['Hebron', 'Jerusalem', 'Nablus',
'Ramallah_Reef']

    for accent in accents:
        folder_path = os.path.join(base_path, accent)
        files = glob.glob(os.path.join(folder_path,
'*.wav'))

        with ThreadPoolExecutor() as executor:
            results = executor.map(extract_features,
files)

        for file_path, feature in zip(files, results):
            if feature is not None:
                features.append(feature)
                labels.append(accent)

    return np.array(features), np.array(labels)

#--------------------------------------------------------#
#-----------------------تقسيم البيانات-----------------------#

# Paths to training and testing data
train_base_path =
'/Users/mahmoudatia/Desktop/Spoken/Train'
test_base_path = '/Users/mahmoudatia/Desktop/Spoken/Test'

# Load and preprocess the datasets
X_train, y_train = load_audio_files(train_base_path)
X_test, y_test = load_audio_files(test_base_path)

# Check if data is loaded correctly
print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")

# Scale the features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


#--------------------------------------------------------#
#--------------------تدريب نموذج SVM----------------------#

# Use Grid Search to find the best parameters for SVM
param_grid = {'C': [0.1, 1, 10], 'kernel': ['linear',
'rbf'], 'class_weight': ['balanced']}
grid_search = GridSearchCV(SVC(), param_grid, cv=5)
grid_search.fit(X_train, y_train)

# Get the best model from Grid Search
best_svm = grid_search.best_estimator_

# Evaluate the model
y_pred = best_svm.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')

# Print confusion matrix and classification report
cm = confusion_matrix(y_test, y_pred,
labels=best_svm.classes_)
```

```
sns.heatmap(cm, annot=True, fmt='d',
xticklabels=best_svm.classes_,
yticklabels=best_svm.classes_, cmap='binary',
linewidths=1)
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

print(classification_report(y_test, y_pred,
target_names=best_svm.classes_))

#--------------------------------------------------------#
#-------------------حفظ النموذج والمقياس----------------------#

# Save the trained model and scaler for future use
with open('best_svm_model.pkl', 'wb') as model_file:
    pickle.dump(best_svm, model_file)
with open('scaler.pkl', 'wb') as scaler_file:
    pickle.dump(scaler, scaler_file)

# Function to predict the accent of a new audio file
def predict_accent(audio_path):
    try:
        y, sr = librosa.load(audio_path, sr=None)
        # Extract MFCCs
        mfccs = librosa.feature.mfcc(y=y, sr=sr,
n_mfcc=20)
        mfccs_mean = np.mean(mfccs, axis=1).reshape(1, -1)
        # Extract Chroma features
        chroma = librosa.feature.chroma_stft(y=y, sr=sr)
        chroma_mean = np.mean(chroma, axis=1).reshape(1, -
1)
        # Extract Spectral Contrast
        contrast = librosa.feature.spectral_contrast(y=y,
sr=sr)
        contrast_mean = np.mean(contrast,
axis=1).reshape(1, -1)
        # Combine all features
        combined_features = np.concatenate((mfccs_mean,
chroma_mean, contrast_mean), axis=1)
        mfccs_scaled = scaler.transform(combined_features)
        predicted_accent = best_svm.predict(mfccs_scaled)
        return predicted_accent[0]
    except Exception as e:
        print(f"Error processing {audio_path}: {e}")
        return None

# Example
test_audio_path =
'/Users/mahmoudatia/Desktop/Desktop/hebron1.wav'
predicted_accent = predict_accent(test_audio_path)
print(f'The predicted accent for the test audio is:
{predicted_accent}')
```