Simple Palestinian Accent Recognition System

Jehad Halahla, Hamza Awashra, Amr Halahla

Electrical and Computer Engineering Dept. Birzeit University
1201467,1201619, 1200902

Abstract

This project aims to utilize machine learning algorithms and spoken language processing techniques to train and test an accent recognition system for the main Palestinian regions: Ramallah-Reef, Nablus, Hebron, and Jerusalem. The primary feature extracted from the provided speech files is the Mel-Frequency Cepstral Coefficients (MFCCs). In preprocessing, the audio files were read, and in some instances, augmented to enhance the dataset. The audio was segmented into 30-second intervals, and 20 MFCCs were extracted from each segment. The data were then scaled and fitted to prepare it for model training. To optimize efficiency, the preprocessed data were stored for reuse, significantly reducing runtime for subsequent operations.
The model we trained is a Support Vector Machine (SVM). We employed grid search to investigate and determine the best parameters, such as kernel type and coefficients. Through this method, we achieved an accuracy of 80%. However, the model faced challenges, particularly with the misclassification of the Nablus accent as other accents, such as those from Hebron and Ramallah-Reef. Despite this, the project demonstrates the potential of using machine learning and MFCCs for regional accent recognition, with room for further refinement and improvement in classification accuracy.

1. Introduction

Accent recognition is crucial for enhancing speech recognition systems and human-computer interactions. This project aims to recognize accents from Palestinian regions: Ramallah-Reef, Nablus, Hebron, and Jerusalem, driven by the cultural and linguistic diversity within Palestine. Accurate accent recognition can improve communication technologies, regional applications, and linguistic studies.

MFCCs (Mel-Frequency Cepstral Coefficients) were chosen for feature extraction due to their effectiveness in capturing speech characteristics. The methodology included preprocessing audio files, augmenting data, segmenting it into 30-second intervals, and extracting 20 MFCCs. A Support Vector Machine (SVM) model was used, optimized via grid search.

Similar methods by other researchers have yielded comparable accuracies for our initial tries, validating our approach. This project aligns with current research and advances the development of region-specific accent recognition systems.

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, and it also discovers how a simple neural network does in terms of metrics.

2.1. Mel-Frequency Cepstral Coefficients (MFCC)

MFCC (Mel-Frequency Cepstral Coefficients) is a key feature extraction method in speech and audio processing. It transforms the audio signal into the frequency domain using the Discrete Fourier Transform (DFT), applies the mel-scale to mimic human hearing, and computes cepstral coefficients. MFCCs highlight important speech features, making them ideal for tasks like speech recognition, speaker identification, and emotion detection [1].

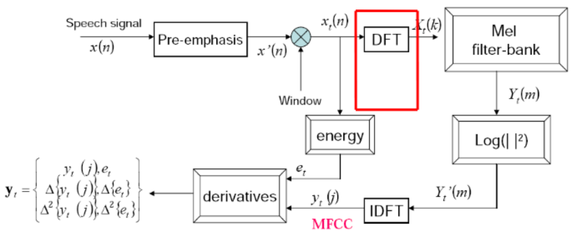

Speech signal
$x(n)$
Pre-emphasis
$x'(n)$
Window
$x_t(n)$
DFT
$X_t(k)$
Mel filter-bank
$Y_t(m)$
Log(| |²)
$Y_t'(m)$
IDFT
$y_t(j)$
MFCC
energy
e_t
derivatives
$\mathbf{y}_t = \left\{ \begin{matrix} y_t(j), e_t \\ \Delta\{y_t(j)\}, \Delta\{e_t\} \\ \Delta^2\{y_t(j)\}, \Delta^2\{e_t\} \end{matrix} \right\}$

Figure 1:roadmap of MFCC extraction [2].

The production of MFCC can be broken down into multiple steps, the steps are as follows:

- **Preemphasis**: Preemphasis boosts the energy magnitude in higher frequencies. In the frequency domain of voiced segments like vowels, higher frequencies have significantly less energy compared to lower frequencies. Enhancing the higher frequency energy improves phoneme detection accuracy, thereby enhancing the overall model performance.

- **Windowing:** The MFCC technique aims to develop the features from the audio signal which can be used for detecting the phones in the speech. But in the given audio signal there will be many phones, so usually when extracting MFCCs, the audio files are segmented as shown in the figure above.

- **DFT (Discrete Fourier Transform):** DFT is an essential part of obtaining the MFCCs, it can be used alone to obtain a lot of the acoustic properties of speech, but it's also used as a necessary step to get the MFCC feature.

- **Mel-Filter Bank**: Human perception of sound differs significantly from machine perception. Our ears are more sensitive to lower frequencies, allowing us to easily distinguish between sounds at 200 Hz and 300 Hz, whereas sounds at 1500 Hz and 1600 Hz are harder to differentiate, despite both pairs having a 100 Hz difference. In contrast, machines maintain the same resolution across all frequencies. Incorporating the characteristics of human hearing into the feature extraction stage can enhance the performance of the model.

- **Applying Log**: Human sensitivity to changes in audio signal energy decreases with higher energy levels. Similarly, the logarithm function reflects this property: higher gradients at lower input values and lower gradients at higher input values. Thus, we apply the logarithm to the Mel-filter output to mimic the human hearing system.

- **IDFT**: In this step, we perform the inverse transform of the previous output. To understand why we need this inverse transform, let's first delve into how sound is produced by humans.

- Sound is generated by the glottis, a valve controlling airflow in the respiratory passages, causing air vibrations. These vibrations occur in harmonics, with the fundamental frequency being the smallest. Other frequencies are multiples of this fundamental frequency. Vibrations then pass into the vocal cavity, which selectively amplifies or dampens frequencies based on tongue and articulator positions, resulting in each sound having a unique articulation.

The below figure shows the signal sample before and after the IDFT operation:

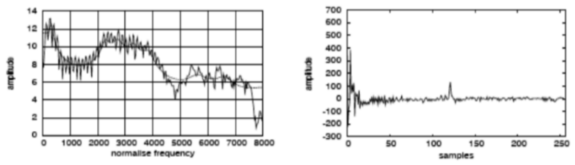


Figure 2: illustration of IDFT [1]

2.2. Support Vector Machine (SVM):

A support vector machine (SVM) is a supervised machine learning algorithm used for classification by finding an optimal hyperplane that maximizes the distance between classes in an N-dimensional space.

Developed by Vladimir N. Vapnik and colleagues in the 1990s, SVMs were introduced in their 1995 paper, "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing." SVMs are effective in classification tasks by finding the hyperplane that maximizes the margin between the closest points of opposite classes. The number of features determines if the hyperplane is a line (2D) or a plane (N-dimensional). Maximizing this margin helps the model generalize well to new data.

SVMs can handle both linear and nonlinear classification. For nonlinearly separable data, kernel functions transform the data into a higher-dimensional space to enable linear separation, known as the "kernel trick." Common kernels include linear, polynomial, radial basis function (RBF), and sigmoid, chosen based on data characteristics and specific use cases [3] [4].

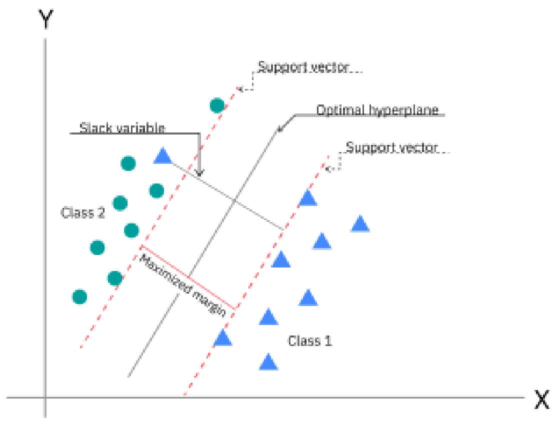


Figure 3: Visualization of SVM [3].

2.2.1. *Linear SVMs*

Linear SVMs are used for linearly separable data, meaning no transformations are needed to separate classes. The decision boundary resembles a "street," with the support vectors acting as the edges. Mathematically, the separating hyperplane is represented as $wx + b = 0$, where w is the weight vector, x is the input vector, and b is the bias term.

There are two approaches to calculating the margin: hard-margin and soft-margin classification. In hard-margin SVMs, data points are perfectly separated by the support vectors, represented by $((wx_j + b)y_j \leq \ a)$. The margin is maximized as $max\, \gamma = a\, ||w||$, where a is the margin. Soft-margin classification introduces slack variables $x_i$allowing for some misclassification. The hyperparameter C adjusts the margin's width: a larger C value narrows the margin for minimal misclassification, while a smaller C value widens it, allowing for more misclassified data.

2.2.2. *Nonlinear SVMs*

Real-world data is often not linearly separable, necessitating nonlinear SVMs. These use preprocessing methods to transform data into a higher-dimensional feature space, which can increase complexity and risk overfitting. The "kernel trick" helps reduce this complexity by replacing dot product calculations with kernel functions, making computation more efficient. Popular kernel functions include the polynomial kernel, radial basis function (RBF) kernel, and sigmoid kernel. These kernels enable SVMs to handle more complex, nonlinear classification tasks effectively.

2.3. Neural Network Multi-Layer Perceptron (MLP Classifier) [5]:

A Multi-layer Perceptron (MLP) is a supervised learning algorithm that learns a function $f: Rm \rightarrow Ro$, where m and o are the dimensions of input and output respectively. Given features $X1, X2, \dots, Xm$ and a target y, it can approximate non-linear functions for either classification or regression. Unlike logistic regression, MLP can have one or more non-linear hidden layers between the input and output layers. This structure is depicted in Figure 1, showing an MLP with one hidden layer and scalar output.

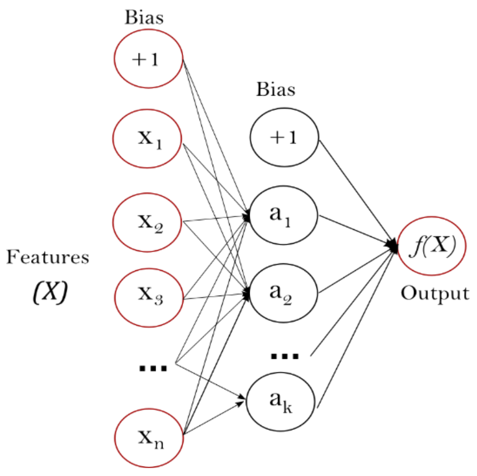


Figure 4: MLP one hidden layer [5]

2.4. Grid Search

Grid search is a technique for fine-tuning hyperparameters in machine learning models, including Support Vector Machines (SVM). Hyperparameters are pre-set configurations that impact the learning process but are not adjusted during training.

In SVMs, two critical hyperparameters are C and gamma:
- The regularization parameter C balances the trade-off between minimizing the decision boundary's complexity and reducing training set error.
- Gamma, a parameter for non-linear hyperplanes, can lead to overfitting if set too high, as it will try to fit the training data too precisely.

Grid search works by generating a grid of possible hyperparameter values and using cross-validation to evaluate model performance for each combination. The term "grid search" comes from its systematic exploration of the grid to find the optimal hyperparameters based on criteria such as accuracy or F1 score.

3. Methodology

In our work we designed 2 different models, they both share the same following steps in order for them to be trained:

3.1. Data preprocessing:

We read the data from the corresponding sub-directories, then in some of our experiments, the audio files were split to 30 second segments in order to increase the training set and to give the SVM and MLP models more data points to work with, we used the librosa python library and pydup, also we used numpy in order to calculate the mean of the mfccs.

3.2. Extracting MFCC:

We used the librosa library in order to calculate the MFCC for the given audio files, calculated the mean for the MFCCs, then appended the result to the feature array, in order for it to be scaled and then fit.

Here are a couple of equations that are used to describe the MFCC extraction process which is depicted in Figure 1:

- Pre-emphasis:

$$y(t) = x(t) - \alpha x(t-1) \text{ [2]}$$

- Framing and Windowing:

$$x_w(n) = x(n) \cdot w(n) \text{ [2]}$$

- Fast Fourier Transform (FFT) and Power Spectrum:

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

- Mel-filterbank:

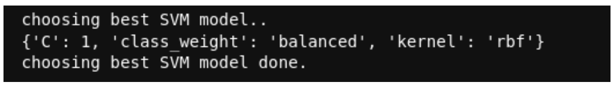
$$H\_m\,(f) = \begin{cases} 0 & if\ f < f_{m-1} \\ \dfrac{f - f_{m-1}}{f_m - f_{m-1}} & if\ f_{m-1} \le f < f_m \\ \dfrac{f_{m+1} - f}{f_{m+1} - f_m} & if\ f_m \le f < f_{m+1} \\ 0 & if\ f \ge f_{m+1} \end{cases} \text{ [2]}$$

- Discrete Cosine Transform (DCT):

$$MFCC(n) = \sum_{K=0}^{K-1} \log\big(X(k)\big) \cos\left(\frac{\pi n(k+0.5)}{K}\right) \text{ [2]}$$

3.3. Feeding the SVM model:

The features were fed to the SVM model, and the parameters for the model were automatically chosen using the grid search technique:



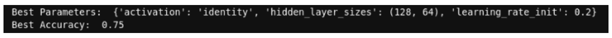
```
choosing best SVM model..
{'C': 1, 'class_weight': 'balanced', 'kernel': 'rbf'}
choosing best SVM model done.
```

Figure 5:best parameters for SVM models

Now the model was ready to do its job, and was able to obtain 80% accuracy in general, which is the best proposed solution.

3.4. Feeding the MLP model

We followed the same steps for this model and were able to determine the best parameters of the neural network:



```
Best Parameters:  {'activation': 'identity', 'hidden_layer_sizes': (128, 64), 'learning_rate_init': 0.2}
Best Accuracy:  0.75
```

Figure 6: best params for the neural network

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 regularization parameter C, the kernel type, and the class weights.

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

The dataset comprises speech samples from four distinct Palestinian regions: Ramallah-Reef, Hebron, Jerusalem, and Nablus. Each region contributed a substantial number of .wav files. Training and testing sets, pre-divided, were loaded separately from assigned directories.

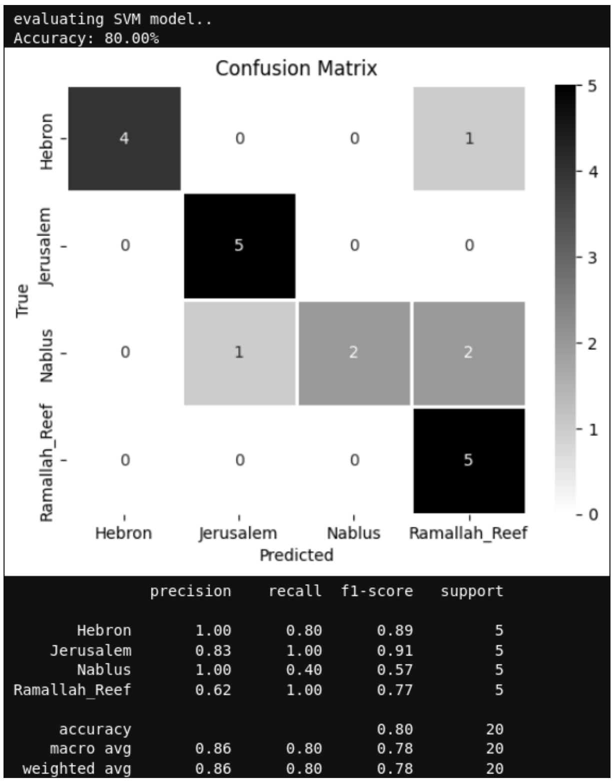
4.4. Results

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

As Figure 7 and Figure 8 show, the SVM model was better in accuracy and many other metrics, and that is based on the limited data, we also tried to segment and augment the data to increase it, the augmentation took a lot of time, so we excluded it. And we sticked to only segmenting it. But it didn’t give the desired results, it only gave 71.4% accuracy for SVM, and around 65% for the MLP, we consider a better estimate of the real accuracy because it virtually included much more data.

We used jupyter notebook for displaying the results, and anything can be tinkered and changed in the code.

4.4.1. SVM model evaluation results:



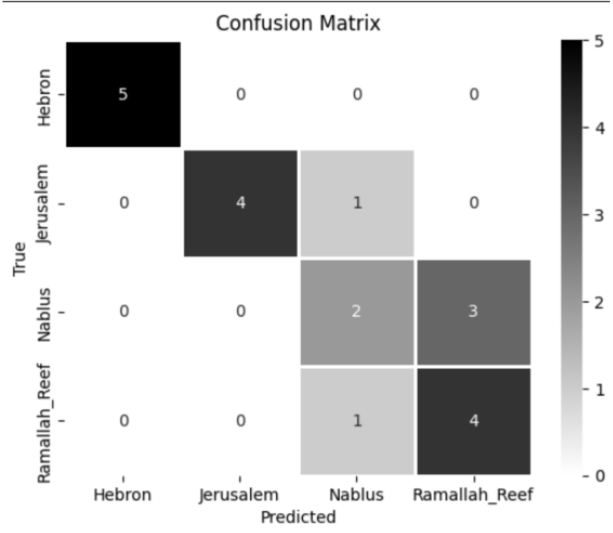
```
evaluating SVM model..
Accuracy: 80.00%
```

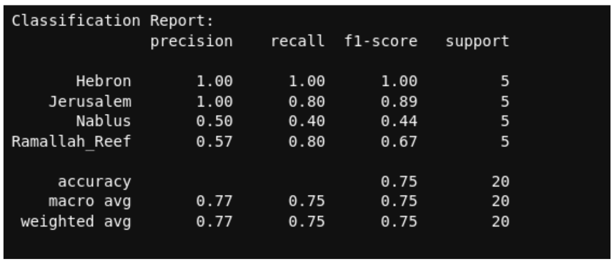
```
               precision    recall  f1-score   support

       Hebron       1.00      0.80      0.89         5
    Jerusalem       0.83      1.00      0.91         5
       Nablus       1.00      0.40      0.57         5
Ramallah_Reef       0.62      1.00      0.77         5

     accuracy                           0.80        20
    macro avg       0.86      0.80      0.78        20
 weighted avg       0.86      0.80      0.78        20
```

Figure 7: confusion matrix, and relevant metrics scores for SVM.





```
Classification Report:
               precision    recall  f1-score   support

       Hebron       1.00      1.00      1.00         5
    Jerusalem       1.00      0.80      0.89         5
       Nablus       0.50      0.40      0.44         5
Ramallah_Reef       0.57      0.80      0.67         5

     accuracy                           0.75        20
    macro avg       0.77      0.75      0.75        20
 weighted avg       0.77      0.75      0.75        20
```

Figure 8: metrics for MLP

5. Conclusion and Future Work

The overall result of our work was satisfying but in the future, we would like to try different combinations of features, and play around more with the parameters, also, we would like to give CNN (Convolutional Neural Networks) a chance, and finally we would refine the test and train data further more to obtain much more realistic results for the evaluations.

6. References

[1] analyticsvidhya. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/. [Accessed 10 6 2024].

[2] "Spoken Language Processing Course slides," 2024.

[3] ibm, "ibm," [Online]. Available: https://www.ibm.com/topics/support-vector-machine#:~:text=What%20are%20SVMs%3F,in%20an%20N%2Ddimensional%20space.. [Accessed 10 6 2024].

[4] H. J. V. J. H. P. P. P. M. C. Saiprasad Duduka, "Accent Classification using Machine Learning," IRJET, 2020.

[5] scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html. [Accessed 10 6 2024].

7. Appendix

The code is on my github repository:
https://github.com/jehad-halahla/accent_identifier