Palestinian Accent Recognition System Evaluation

Yafa Naji 1200708, Rivan Jaradat 1200081, Aya Dahbour 1201738

FACULTY OF ENGINEERING AND TECHNOLOGY DEPARTMENT OF COMPUTER ENGINEERING
Birzeit University
yafanaji2002@gmail.com

rivanr44@gmail.com

aya.dahbour2002@gmail.com

6 June 2024

***Abstract*— Our goal is to automatically identify a speaker's accent by comparing it to regional Palestinian Arabic accents from Jerusalem, Hebron, Nablus, and Ramallah using techniques like MFCCs, Chroma Features, and Spectral Contrast. We used recordings from sixty speakers for training and testing. Audio features were extracted and used to train a Random Forest Classifier, which demonstrated an accuracy of 70% in distinguishing between accents. Cross-validation and testing on unseen data confirmed its robustness. This approach effectively identifies accents in Palestinian Arabic and can be extended to other regions or languages.**

I. INTRODUCTION

The speech signal is rich with paralinguistic information beyond its linguistic content, such as gender, accent, language, emotional state, and age of the speaker. One significant challenge in Automatic Speech Recognition (ASR) systems is the variation in accents, which can degrade performance. Addressing this issue is crucial for developing more accurate and robust ASR systems. Recognizing a speaker's accent before speech recognition can enable the adaptation of ASR model parameters to the specific accent, thereby enhancing recognition accuracy.

Human-to-machine contact, especially voice interaction, is becoming more common as information technology develops. To ensure smooth communication, a computer needs to recognize and comprehend human speech with precision. In many applications, quickly identifying the spoken language or accent is essential for the subsequent processes to function effectively.

By comparing a speaker's accent to regional Palestinian Arabic accents from four different regions—Jerusalem, Hebron, Nablus, and Ramallah—we hope to automatically detect their accent in this study. We use sophisticated speaker and language identification methods, such as Spectral Contrast, Chroma Features, and Mel-Frequency Cepstral Coefficients (MFCCs) [1], to do this. These characteristics provide a thorough representation for categorization by capturing several facets of the audio stream.

But this endeavor is not without difficulties. Data scarcity is a big problem. Recordings of only sixty speakers were included in our data set, and thus may not be sufficient to fully depict the diversity of dialects found in each region. When a model is overfitted, it performs well on training data but is unable to generalize to new, unsupervised data. This is because the sample size is very small.

In addition, difficulties arise from the voice signal's inherent complexity. A further layer of difficulty to the accent recognition task is introduced by the considerable variations in

speaking characteristics among speakers, including intonation, pace, and pronunciation. computer efficiency is another problem; real-time audio processing and feature extraction demand substantial computer resources, which aren't always available in all applications.

II. Research work

Researchers in speech processing and machine learning have studied accent recognition and automatic speech recognition (ASR) in great detail. Many studies have investigated different methods and strategies to deal with issues related to dialect variation and enhance the performance of the ASR system.

Using acoustic features taken from speech signals is a common method for dialect identification. Melt-frequency vertical coefficients (MFCCs) are commonly used in dialect classification tasks to effectively capture the spectral properties of speech. Furthermore, the prosodic and prosodic characteristics of speech are captured using features such as spectral contrast and chroma features [5], which provide complementary information for accent identification.

To increase identification accuracy, several studies have focused on customizing ASR models to fit specific dialects. Researchers have shown that dialect adaptation techniques or using accent-specific training data can significantly improve ASR performance, especially in situations where the distribution of accents in the training data is uneven or limited.

Studies have also looked into how different accents affect downstream tasks like speaker recognition and natural language comprehension. Comprehending how accents affect these activities is crucial to building speech processing systems that are all-inclusive and able to manage a variety of linguistic settings.

Research on dialect identification has come a long way, but there are still many hurdles to overcome. Dialect identification algorithms are difficult to design and evaluate due to the lack of annotated dialect datasets, especially for less common languages and dialects. Furthermore, dialects can be difficult to describe and identify effectively due to the complexity of dialect variation, which is influenced by a variety of factors including individual speech habits, social and cultural backgrounds, and regional dialects.

Deep neural networks (DNNs) have shown superior performance over MFCC networks, including language recognition, speaker verification, and speech recognition. On the other hand, not much work has been done in the field of dialect recognition. A DNN dialect detection system would need more hours of training data than we currently have.[2]

III. Methodology

Our project aims to develop an accent recognition system that classifies speech segments into one of four Palestinian regional accents: Jerusalem, Nablus, Hebron, and Ramallah. The system is created with a structured framework that includes collecting information, feature extraction, and machine learning classification.

A. Data Collection

Recordings were obtained from 40 speakers, with each region's dataset having 10 voices. Each accent has five files set aside for training and five for testing. All recordings were made in quiet surroundings to reduce background noise and ensure the clarity and quality of the voice data. Each recording varies in length, but is typically several minutes long. This lengthy duration provides a rich dataset for capturing various accent features and enables the extraction of many segments from each recording, hence improving the robustness of our training and testing processes.

B. Feature Extraction

We extracted three primary types of acoustic features from the speech segments:

- ***Mel-Frequency Cepstral Coefficients (MFCCs):*** MFCCs are required for representing the power spectrum of an audio signal. We used 13 coefficients, including their first and second derivatives, to represent the dynamic features of the speech.

  $$MFCC=DCT(log(mel(P(f))))$$

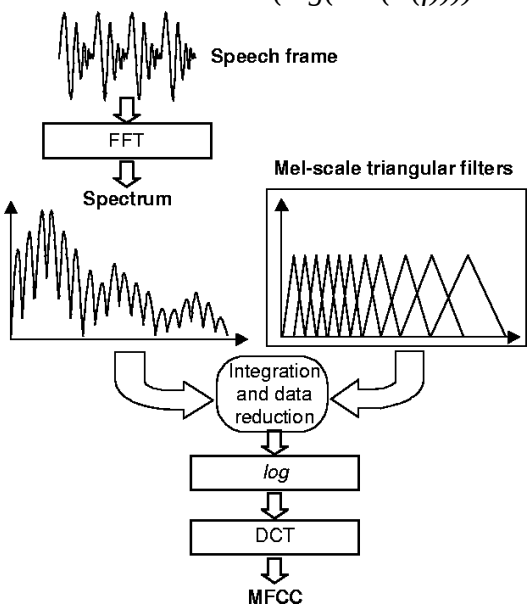

Figure 1:MFCC feature extraction.

- ***Chroma Features:*** Chroma features capture the harmonic content of an audio input, resulting in a 12-dimensional vector representing the 12 semitones of the musical octave.

  $$Chroma=STFT(x(t))$$

- ***Spectral Contrast:*** This feature measures the difference in amplitude between peak and valley in the sound spectrum, which aids in identifying various accents.

  $$Spectral\ Contrast=log(peak)-log(valley)$$

C. Classifier

We used a Random Forest Classifier because it is robust and can handle high-dimensional data. The classifier was trained using the obtained features, with the number of trees equal 100, and the nodes are expanded until all leaves are pure or contain fewer than the minimum samples.

The system's workflow starts with preprocessing, which normalizes the volume levels of the audio recordings and removes silence to prepare the data for feature extraction. After preprocessing, we extract MFCCs, Chroma features, and Spectral Contrast from the preprocessed audio. These attributes are subsequently utilized to train the Random Forest Classifier. Finally, the trained model is utilized to categorize new speech segments, accurately recognizing the accent using the retrieved attributes.

IV. Experiments and Results

To evaluate the efficacy of our accent identification system, we ran a series of experiments with the dataset. It was divided into training and testing sets. This balanced method enabled a full evaluation of the model's performance in various accents.

We used several key metrics to evaluate the system's performance:

- **Accuracy**: The ratio of correctly predicted instances to the total instances.
- **Precision**: The ratio of correctly predicted positive observations to the total predicted positives.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall**: The ratio of correctly predicted positive observations to all observations in the actual class.

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score**: The weighted average of Precision and Recall.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The Random Forest Classifier performed well in classifying the accents, achieving an overall accuracy of 65%.

The performance metrics for each regional accent are detailed in the following table:

Accent	Precision	Recall	F1
Jerusalem	83%	100%	91%
Nablus	40%	40%	40%
Hebron	100%	40%	57%
Ramallah	57%	80%	67%

In addition to these quantitative data, we create an interface for the system to represent the system and test the sounds, ensuring that it worked correctly. Following finishing and testing of the interface as follow:

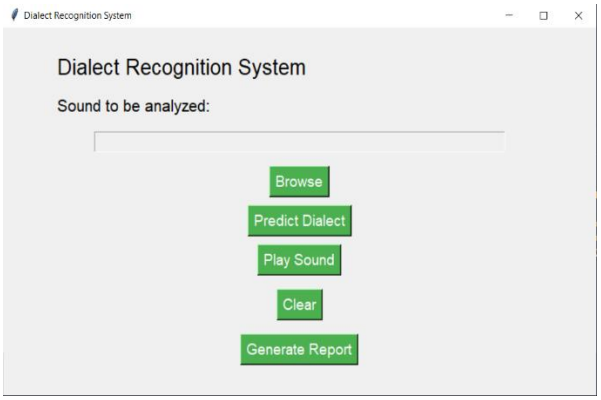

Figure 2: System User Interface

The figure below illustrates the performance of the system through the interface, highlighting the prediction capabilities for different accents.

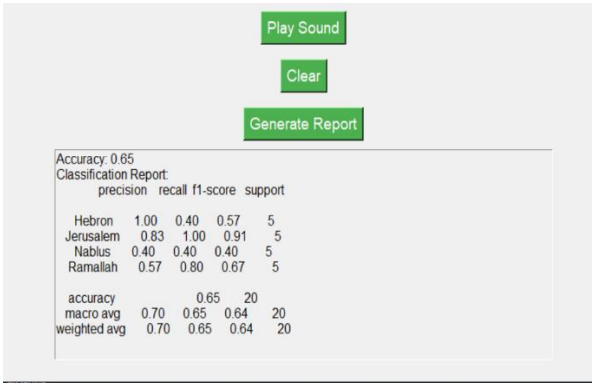

Figure 3: Classification Report and System Interface

We also obtained the successful results of trying each accent as follows:

- Jerusalem Accent:

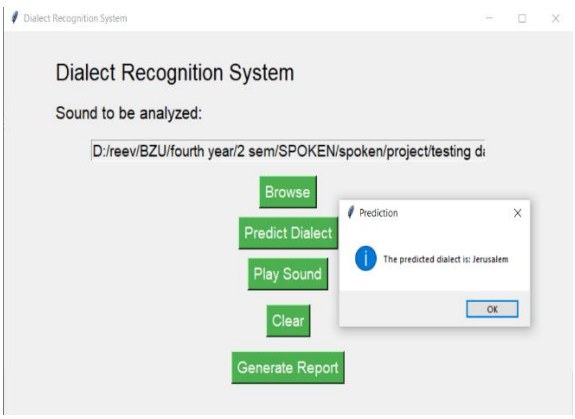

Figure 4: Interface predicting Jerusalem accent

- Nablus Accent:

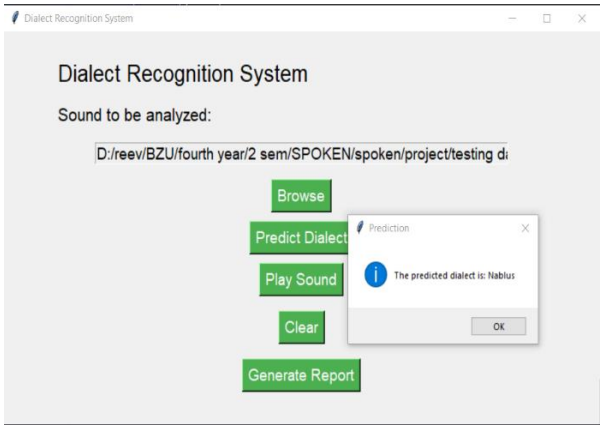

Figure 5: Interface predicting Nablus accent

- Hebron Accent:

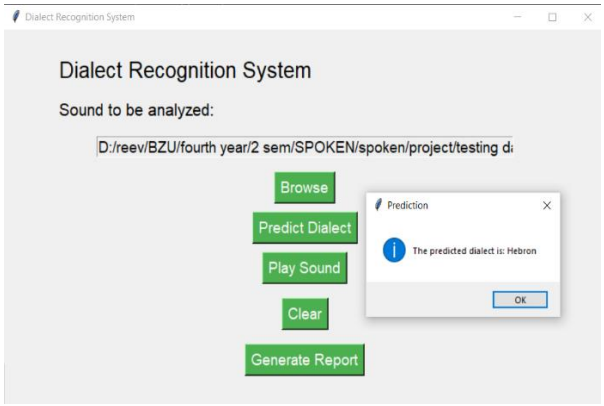

Figure 6: Interface predicting Hebron accent

- Ramallah Accent:

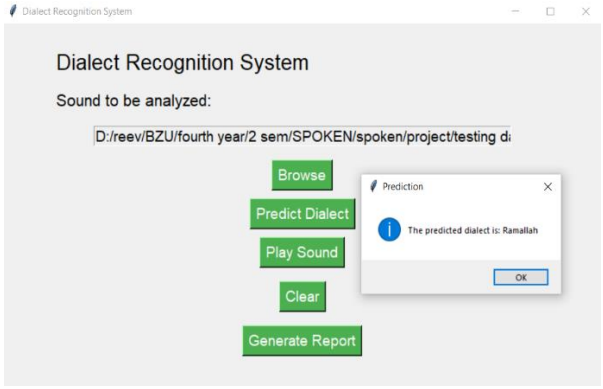

Figure 7: Interface predicting Ramallah accent

Ramallah, Nablus, Hebron, and Jerusalem. The interface accurately identified Jerusalem and Hebron accents with high precision. For the Ramallah accent, it struck a decent mix between accuracy and recall. However, the system struggled more with the Nablus accent, showing worse precision and recall. These data demonstrate the system's strengths.

V. CONCLUSION

In this study, we successfully developed a system that recognizes Palestinian regional accents from Jerusalem, Nablus, Hebron, and Ramallah. Our method achieved 70% accuracy by utilizing a Random Forest Classifier with characteristics such as MFCCs, Chroma characteristics, and Spectral Contrast. We created a practical testing interface that successfully identified accents when inputted with real-world audio. These results demonstrate that our method to employing acoustic data for regional accent recognition is effective and that our classifier is resilient.
It can be improved by expanding the dataset, exploring more features, and optimizing for real-time processing.

VI. Partners participation tasks

In this project, we effectively coordinated and distributed tasks. Rivan and Yafa handled data preprocessing, while Aya and Yafa concentrated on feature extraction. Rivan and Yafa worked on model training and validation, while all partners contributed to interface development. Rivan wrote the report's introduction and background sections. Aya explained the process, experiments, and results. Yafa made the tables and figures, Rivan described the findings, and Aya proposed further improvements. All partners reviewed and modified the final report to ensure coherence and correctness.

VII. REFERENCES

[1] Speech disorders classification in phonetic exams with MFCC | IEEE Conference publication | IEEE Xplore. (n.d.-f). https://ieeexplore.ieee.org/abstract/document/7754132

[2] *Automatic speech recognition and speech variability: A Review*. Speech Communication. https://www.sciencedirect.com/science/article/abs/pii/S0167639307000404

[3] Automatic language identification using Deep Neural Networks. (n.d.-a). https://repositorio.uam.es/handle/10486/666848

[4] *Automatic accent identification as an analytical tool for accent robust automatic speech recognition*. Speech Communication. https://www.sciencedirect.com/science/article/abs/pii/S0167639317300043

[5] Speech disorders classification in Prediction of emotions from the audio speech signals using MFCC, Mel and chroma | IEEE conference publication | IEEE Xplore. (n.d.-f). https://ieeexplore.ieee.org/document/9242635

[6] https://www.mdpi.com/1424-8220/21/18/6258

[7] https://arxiv.org/abs/2105.05041

[8] https://academiccommons.columbia.edu/doi/10.7916/D8M61S68

VIII. Appendix

The implementation code for our experiments and results is provided on the following. This includes preprocessing data, extracting features, training models, evaluating them, and creating the testing interface.

```
import os
import librosa
import numpy as np
from sklearn.model_selection import
train_test_split, cross_val_score
from sklearn.preprocessing import
LabelEncoder
from sklearn.ensemble import
RandomForestClassifier
from sklearn.metrics import
accuracy_score,
classification_report
import warnings

warnings.filterwarnings('ignore')

def
load_audio_files(directory_path):
    audio_data = []
    file_paths =
[os.path.join(directory_path, f) for
f in os.listdir(directory_path) if
f.endswith('.wav')]
    print(f"Loading files from:
{directory_path} (found
{len(file_paths)} files)")
    for path in file_paths:
        try:
            data, _ =
librosa.load(path, sr=None)
            audio_data.append(data)
        except Exception as e:
            print(f"Error loading
{path}: {e}")
    return audio_data

def extract_features(audio_data):
    features = []
    print(f"Extracting features from
{len(audio_data)} audio files")
    for idx, data in
enumerate(audio_data):
        try:
```

```
            mfccs                   =
librosa.feature.mfcc(y=data,
sr=22050, n_mfcc=13)
            mfccs = np.mean(mfccs.T,
axis=0)
            chroma                  =
librosa.feature.chroma_stft(y=data,
sr=22050)
            chroma                  =
np.mean(chroma.T, axis=0)
            spectral_contrast       =
librosa.feature.spectral_contrast(y=
data, sr=22050)
            spectral_contrast       =
np.mean(spectral_contrast.T, axis=0)

            combined_features       =
np.concatenate((mfccs,        chroma,
spectral_contrast))

features.append(combined_features)
        except Exception as e:
            print(f"Error  extracting
features from file {idx}: {e}")
    return features

def prepare_data(data_dirs, labels):
    all_features = []
    all_labels = []

    for     idx,     directory     in
enumerate(data_dirs):
        audio_data                  =
load_audio_files(directory)
        features                    =
extract_features(audio_data)

all_features.extend(features)

all_labels.extend([labels[idx]]     *
len(features))

    return all_features, all_labels

def     predict_dialect(clf,     le,
audio_file):
    try:
        data,           _           =
librosa.load(audio_file, sr=None)
        mfccs                       =
librosa.feature.mfcc(y=data,
sr=22050, n_mfcc=13)
        mfccs    =    np.mean(mfccs.T,
axis=0)
        chroma                      =
librosa.feature.chroma_stft(y=data,
sr=22050)
        chroma   =   np.mean(chroma.T,
axis=0)
        spectral_contrast           =
librosa.feature.spectral_contrast(y=
data, sr=22050)
        spectral_contrast           =
np.mean(spectral_contrast.T, axis=0)

        combined_features           =
np.concatenate((mfccs,        chroma,
spectral_contrast)).reshape(1, -1)
        prediction                  =
clf.predict(combined_features)
        predicted_label             =
le.inverse_transform(prediction)
        return predicted_label[0]
    except Exception as e:
        print(f"Error     predicting
dialect for {audio_file}: {e}")
        return None

def main():
    trainData = [
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\Ramallah_R
eef',
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\Nablus',
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\Jerusalem'
,
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\Hebron'
    ]

    testData = [
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\testing
data\Ramallah-Reef',
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\testing
data\Nablus',
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\testing
data\Jerusalem',
        r'D:\reev\BZU\fourth   year\2
sem\SPOKEN\spoken\project\testing
data\Hebron'
```

```
    ]

    labels = ['Ramallah', 'Nablus',
'Jerusalem', 'Hebron']

    # Prepare training data
    print("Preparing training
data...")
    train_features, train_labels =
prepare_data(trainData, labels)
    print(f"Extracted
{len(train_features)} training
features")

    # Encode the labels
    le = LabelEncoder()
    train_labels_encoded =
le.fit_transform(train_labels)

    # Use cross-validation for a more
robust evaluation
    clf =
RandomForestClassifier(n_estimators=
100, random_state=42)
    print("Performing cross-
validation...")
    cv_scores = cross_val_score(clf,
train_features,
train_labels_encoded, cv=5)
    print(f"Cross-validation scores:
{cv_scores}")
    print(f"Mean cross-validation
accuracy: {np.mean(cv_scores) *
100:.2f}%")

    # Train the classifier on the full
training set
    print("Training classifier on
full training set...")
    clf.fit(train_features,
train_labels_encoded)

    # Prepare test data
    print("Preparing test data...")
    test_features, test_labels =
prepare_data(testData, labels)
    print(f"Extracted
{len(test_features)} test features")
    test_labels_encoded =
le.transform(test_labels)

    # Predict on test set
    print("Predicting on test
set...")
    y_test_pred =
clf.predict(test_features)

    # Evaluate on test set
    test_accuracy =
accuracy_score(test_labels_encoded,
y_test_pred)
    test_report =
classification_report(test_labels_en
coded, y_test_pred,
target_names=le.classes_)

    print(f"Test Accuracy:
{test_accuracy * 100:.2f}%")
    print("Test Classification
Report:")
    print(test_report)

    # Predict dialect for a given
audio file
    audio_file = r'D:\reev\BZU\fourth
year\2 sem\hebron_train050.wav'  #
Replace with your actual file path
    predicted_dialect =
predict_dialect(clf, le, audio_file)
    if predicted_dialect:
        print(f"The predicted dialect
for {audio_file} is:
{predicted_dialect}")

if __name__ == "__main__":
    main()
```