Speech Emotion Detection

Semester Project
Session 2023-2027
BS in Artificial intelligence

Department of Software Engineering Faculty of Computer Science & Information Technology The Superior University, Lahore

Name: M.Junaid

Roll no: BSAI-109

SECTION: 3B

Table of Contents

1. Project Overview

The goal of this project is to build a Speech Emotion Detection system using machine learning techniques to identify emotions in speech based on audio files. The system uses a dataset of speech samples, extracts features, and trains a model to classify emotions. The final model is then used in a graphical user interface (GUI) application where users can upload audio files, and the system predicts the emotion conveyed in the speech.

2. Data Collection and Preprocessing

- **Dataset**: The project uses the "Tess" dataset, which contains various speech samples in different emotional states. The dataset is read by traversing through the directory and extracting the file paths and associated labels (emotion).

  Code Snippet for Data Collection:

```
for dirname, _, filenames in os.walk('Tess'):
    for filename in filenames:
        paths.append(os.path.join(dirname, filename))
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())
```

- **Feature Extraction**: For each audio file, the system extracts Mel-frequency cepstral coefficients (MFCCs), which represent the speech characteristics that are most likely to capture emotional cues. The features are averaged across time to create a feature vector for each file.

  Code Snippet for Feature Extraction:

```
def extract_features(file_path):
    y, sr = librosa.load(file_path, sr=None)
    mfcc = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13)
    mfcc_mean = np.mean(mfcc, axis=1)
    return mfcc_mean
```

3. Model Training

- **Training Data**: The extracted features are used to create a DataFrame, and the labels (emotions) are associated with the feature vectors. The dataset is split into training and test sets using an 80-20 split.
- **Model Selection**: A Support Vector Machine (SVM) with a linear kernel is chosen as the model for classification. SVMs are effective for high-dimensional data, which makes them suitable for audio feature classification.

  Code Snippet for Model Training:

```
model = SVC(kernel='linear')
model.fit(X_train, y_train)
```

  Evaluation: The model's performance is evaluated using the `classification_report`, which shows metrics such as precision, recall, and F1-score for each class.

  Code Snippet for Evaluation:

```
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```

- **Saving the Model**: After training, the model is saved using `pickle` for future use in the GUI application.

  Code Snippet for Saving the Model:

```
with open('emotion_speech_recognition_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

4. Graphical User Interface (GUI)

- **GUI Framework**: The GUI is built using Tkinter, which allows users to interact with the emotion recognition system easily.
- **File Upload**: Users can upload audio files (in .wav, .mp3, or .flac formats). The selected file is then passed to the model for emotion classification.
- **Predictions**: The predicted emotion is displayed on the GUI, and the result is printed to the console as well.

Code Snippet for GUI Implementation:

```
def upload_audio_file():
    file_path = filedialog.askopenfilename(filetypes=[("Audio Files",
    if file_path:
        classify_audio_file(file_path)
    else:
        messagebox.showwarning("No File", "Please select a valid audio
```

Main Window: The main window has a button to upload the audio file and a label to display the predicted emotion.

Code Snippet for GUI Layout:

```
root = tk.Tk()
root.title("Emotion Recognition from Audio File")
root.geometry("400x300")


result_label = tk.Label(root, text="Predicted Emotion: ", font=("Helvetica", 14))
result_label.pack(pady=20)


upload_button = tk.Button(root, text="Upload Audio File", font=("Helvetica", 14), command=upload_audio_file)
upload_button.pack(pady=20)
```

5. Conclusion

- **Results**: The trained model is capable of classifying emotions in speech based on audio features extracted using MFCC. The GUI provides a simple way for users to interact with the system and receive emotion predictions from uploaded audio files.
- **Future Work**: The system can be expanded by integrating more advanced features for speech emotion recognition, exploring other machine learning models, and incorporating real-time emotion detection.