Speech Emotion Recognition

Business Objective

Emotions are incredibly vital in the mental existence of humans. It is a means of communicating one's point of view or emotional state to others. Humans can sense the emotional state of one another through their sensory organs. Whereas doing the same for a computer is not an easy task. Although computers can quickly comprehend content-based information, obtaining the depth underlying content is challenging, which is what speech emotion recognition aims to accomplish.

The extraction of the speaker's emotional state from his or her speech signal is known as Speech Emotion Recognition (SER). There are a few universal emotions that any intelligent system with finite processing resources can be trained to recognize or synthesize as needed. These include neutral, calm, fear, anger, happiness, sad, etc. Speech emotion recognition is being more widely used in areas such as medical, entertainment, and education.

In this project, we will build a model that will be able to recognize emotions from sound files with the help of keras and tensorflow libraries. We will also build a model using an MLP from sklearn library.

Data Description

The dataset in use is the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS). It contains a total of 7356 files. Two lexically-matched statements are vocalized in a neutral North American accent by 24 professional actors (12 female, 12 male) in the database. Calm, happy, sad, angry, afraid, surprise, and disgust expressions can be found in speech, whereas calm, happy, sad, angry, and fearful emotions can be found in song. Each expression has two emotional intensity levels (normal and strong), as well as a neutral expression. All three modalities are available: audio-only (16bit, 48kHz.wav), audio-video (720p H.264, AAC 48 kHz, .mp4), and video-only (720p H.264, AAC 48 kHz, .mp4) (no sound).

For this particular project we will be making use of the audio-only files. Audio-only files of all actors (01-24) are available as two separate zip files (~200 MB each):

- Speech file (Audio_Speech_Actors_01-24.zip, 215 MB) contains 1440 files: 60 trials per actor x 24 actors = 1440.
- Song file (Audio_Song_Actors_01-24.zip, 198 MB) contains 1012 files: 44 trials per actor x 23 actors = 1012.

Each of the RAVDESS files has a unique filename. The filename consists of a 7-part numerical identifier (e.g., 02-01-06-01-02-01-12.mp4). These identifiers define the stimulus characteristics. Filename identifiers are as follows-

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).

- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Aim

To build an artificial neural network model that is able to correctly classify speech audio files into different emotions such as happy, sad, anger, neutral, etc.

Tech Stack

- Language - Python
- Libraries – Keras, tensorflow, librosa, soundfile, sklearn, pandas, matplotlib, numpy, pickle

Approach

1. Importing the required libraries and packages
2. Open the config.ini file. (This is a configuration file which can be edited according to your dataset)
3. Read the dataset (audio files)
4. Visualize the structure of a sound file
5. Understand the features of a sound file
6. Extract features
7. Load the entire data
8. Train-Test Split
9. Model Training
    - Using Keras and Tensorflow
    - Using MLP from scikit-learn
10. Hyperparameter optimization
11. Code modularization for production

Modular code overview

```
input
|_config.ini
|_ Audio_Song_Actors_01-24
|_ Audio_Speech_Actors_01-24

src
|_engine.py
|_ml_pipeline
        |_utils.py
        |_model.py

lib
|_SpeechEmotions.ipynb

output
|_keras
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. input

2. src

3. output

4. lib

1. input folder **-** It contains all the data that we will need for analysis.
    - A config file, with some basic configuration parameters which can be edited according to your dataset.
    - An Audio_Song_Actors_01-24 folder. This folder consists of audio song samples of all 24 actors.
    - An Audio_Speech_Actors_01-24 folder. This folder consists of audio speech samples of all 24 actors.

2. src folder - This is the most important folder of the project. This folder contains all the modularized code for all the above steps in a modularized manner. This folder consists of:
    - engine.py
    - ml_pipeline

The ml_pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the engine.py file.

3. output folder **–** The output folder contains the best fitted model that we trained for this data. This model can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
4. lib folder **-** This is a reference folder. It contains the original ipython notebook that we saw in the videos.

Project takeaways

1. Understanding the problem statement
2. Understanding basic architecture of a neural network
3. Understanding structure of sound waves
4. Understanding Fourier Transform
5. Understanding the file structure of the data
6. Learning visualization of sound waves using spectrograms
7. Understanding zero-crossing
8. Using libraries like librosa and soundfile
9. Using libraries like matplotlib, pandas, numpy, etc
10. Importing the audio files and playing them
11. Model training using Keras and Tensorflow
12. Model training using scikit-learn
13. Using MLP model
14. Creation of config files
15. Understanding Hyperparameter optimization
16. Learning how to save models
17. Code Modularization