Voice Commands Based Ordering System

This hackathon has been designed to help you practice, reinforce and apply various concepts learned in Module - 1.

Objective:

Upon successful completion of this Hackathon, you will integrate a classifier, which can classify the voice commands, into a food ordering system.

Data:

The data contains voice samples of (classes) 'Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine', 'yes' and 'no'. Each of class is denoted by a numerical value in the following way.

Class of samples	Labels
Zero	0
One	1
Two	2
Three	3
Four	4
Five	5
Six	6
Seven	7
Eight	8
Nine	9
Ten	10
Eleven	11

- The audio files recorded in the studio are saved with the following naming convention:
  - Class Representation + user_id + counter
  - For example: The 35th voice sample by the user b2, which is "Yes", is saved as 10_b2_35.wav.
  - Here 10 will be the label of that sample
- The audio files recorded in a noisy environment with built in noise cancellation microphone are saved with following naming convention
  - Class Representation + "n" + user_id + counter.
  - For example: The 38th voice sample by the user g3, which is "no", is saved as 11_g3_38.wav.
  - Here 11 will be the label of the sample.

Installation Steps:

Steps to install Spyder:
`https://pythonhosted.org/spyder/installation.html`
Steps to install Pytorch:
Go to the link given below:
`https://pytorch.org/`

1. Choose the appropriate OS.
2. Choose either to install with conda (which needs Anaconda to be installed in your system) or pip or Source (to directly download from GitHub).
3. Choose the python version (acc. to python version installed in your system).
4. Choose CUDA as None, if you don't gpu in your system

Run the command obtained to install PyTorch.
Please find the steps below different OS : Windows, Mac and Ubuntu.
//python3 and pip3 need to be installed
MAC:
Need HomeBrew to install portaudio
Run the following command from the terminal to install Homebrew:
/usr/bin/ruby -e "$(curl –fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

Commands to install necessary to packages:
brew install portaudio
pip3 install pyaudio
pip3 install keyboard
pip3 install librosa

Ubuntu:
sudo apt-get install -qq -yy portaudio19-dev
sudo apt-get install python3-pyaudio
pip3 install keyboard
pip3 install librosa

Windows: (needs Anaconda to install portaudio)
pip3 install pyaudio
conda install -c anaconda portaudio
pip3 install keyboard
pip3 install librosa

Python Files present:

- Record_audio.py
- Order.py
- Collect_voice_commands.py
- Interactive_voice_sample_collector.py

Record_audio.py:

- get_features(): (Code is fully implemented/given)
  - Parameters:
    - filepath: path of the audio file
    - sr(samplingrate = 8000) : for all the recordings given and newly recorded audio files using the program are recorded with a sampling rate of 8000
    - n_mfcc: 30
    - n_mels: 128
    - frames: 1
  - Details:
    - returns deep learned features, given the file path of the audio
- get_network(): (Code is fully implemented/given)
  - Parameters : NA
  - Details:
    - This function return a deep neural network for deep features used in the get_features() function
- record_voice() : (Code is fully implemented/given)
  - Parameters:
    - Username: an id given to a person
    - j: label of the sample
    - v: number of the sample recorded by a person with a particular ID
  - Details:
    - This function when invoked records the audio from the system's default microphone for a second

- play_audio():(Code is fully implemented/given)
  - Parameters:
    - Path: path of the audio file to be played
  - Details:
    - This function plays an audio (.wav) file given it path
- Plotchart():(Code is fully implemented/given)
  - Parameters:
    - Objects: list of elements on X-axis
    - Confidence: list of confidence values respectively for corresponding objects
  - Details:
    - Plots bar chart for each and every sample vs its latest confidence metric

Order.py: (Code partially implemented/ training classifier has to be coded by you)

Class order ():
Methods:
prompt_menu():
Parameters:

- list_task: List of lists (1st list contains the menu of the food, 2nd list contains the quantity/servings)
- flag: to track whether the input taken is for menu or quantity

classify_input(): (You have to Complete this function so that it returns predicted label and confidence measure in the same order)
Parameters:

- Define your own parameters, as per your classifier model

Details:

- While ordering the food, this method has to classify the input voice sample

confirm_input(): (Code is fully implemented/given)
Parameters:

- digit: predicted label for the given voice sample
- confidence: confidence measure obtained from the classify_input() function's output
- flag : to track the menu and quantity

take_user_input(): (Code is fully implemented/given)
Parameters:

- list_task
- flag

Details:

- It prompts for the menu/quantity, records the value using record_voice function in Record_audio.py files and extracts features from getfeatures() function in Record_audio.py files
- By passing these extracted features to classify_input() method, you have to predict the label and obtain a confidence measure
- Then the confidence measure and predicted digit are passed to confirm_input() function to check whether the confidence measure is high enough. If the confidence is high enough and predicted label is valid then this function returns the menu choice and label, otherwise calls the take_user_input() again to start the whole process once more

Collect_voice_commands.py (code fully implemented):

- Details:
  - When the program is executed, it prompts to record the voice and saves the files with the naming convention given in the document

classifier_training.py (code to be implemented by you):

- Use this file to train your classifier

Interactive_voice_sample_collector.py: (Code partially implemented/ training per sample has to be coded by you)

- Details:
  - When the program is executed, it prompts to record the voice
  - Once recorded, plays back the sample immediately
  - Extract features using the getfeatures() function in the file Record_audio.py and update/modify the classifier, which is implemented by you using the samples given to you.
  - Code to update/modify your classifier has to be implemented by you.

Tasks:

Stage 1:

- Build a classifier to classify the voice samples given in the speech_data correctly using the features from get_features() function in Record_audio.py file.
- Max. Marks : 15
  - Score you get : your classifier accuracy percentage of 15
  - Example : If a team gets, 80 % accuracy on the test set, then the marks will be 80% of 15 marks i.e. 12 marks (will round of the score, in case of non - integer scores).

Stage 2:

- Team has to collect the voice commands by executing the "Collect_voice_commands.py".
- The id should be "t"+team_number + "n" identifier for each team member + gender(m, f).
  - Example : if team 49's 4th member, who is male, is recording. Then the identifier should be t49n4m.
  - Example : if team 50's 3th member, who is female, is recording. Then the identifier should be t50n3f.
- Collected voice samples should be upload as a zip file to Link google drive.

Stage 3:

- Now using your samples, refine / retrain the classifier you implemented in stage 1
- Use this classifier in the Order.py and use it to classify your response in the method classify_input().
- They are two stages in ordering
  - Ordering the item correctly
  - giving the input quantity correctly
- Max. Marks : 10, when both the steps are done correctly (sample voice from team member)

Stage 4:

- Refine/train the classifier with all the available data (voice samples) which is present in the above mentioned google drive link
- Place an order, Similar to stage 3, but the voice sample will be given by the evaluationer
- Max. Marks : 5, upon correctly classifying both the item and quantity

Bonus:(Max. Marks : 10)

Implement any one of it:(Only one bonus will be considered)

- Introduce new vocabulary other than those mentioned above
- Train your classifier online for every sample you record