|  |
| --- |
| **CMSC5707 Assignment 3, submission deadline: 23:59, 5 Dec 2021**  (Late submission penalty: 15% marks reduction of this assignment for each 24 hours)  Please zip all your files into one zip file. Submit your zip file to CUHK Blackboard system  (<https://blackboard.cuhk.edu.hk/>)  and follow the naming format: [Asg-X][SID][Name].zip  For example, [Asg-3][1234567890][Chan Tai Man].zip |

Programming Question (35% of assignment 3)

1. Build a small vocabulary speech recognition system

Build a small vocabulary speech recognition system using LSTM (Long Short Term Memory) neural network model. Run the Music genre classification demo program shown in (b) first. Then you can build your system based on this framework.

1. **Aim and Objective**

To learn how to build a small speech recognition system using the LSTM model based on the Tensor-flow programing framework.

1. **Tools installation:** Follow the instruction to install the tools if necessary.
   * Goldwav: <https://www.goldwave.com/>, it is required to record and edit sound files.
   * Tensorflow: [hong'slinks - 1.Tensor\_windows (installation through Anaconda3) (google.com)](https://sites.google.com/site/hongslinks/1-tensor_windows-installation-through-anaconda3?authuser=0)
   * Music genre classification in <https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification>, Run this first, then you can build your system based on this framework
   * Librosa: <https://librosa.org/> (convert .au sound files to MFCC code)
2. **Requirements and data preparation:**
   * You need to recognize 5 different words e.g. NAMES of your friends, such as Peter Goldsmith; Tom smith; Mary Goodman etc. It can be spoken in English, Cantonese, or Mandarin. These names can also be countries or objects.
   * The duration of the recording sound for each NAME should last not less than 3.1 seconds. Because the original software assume each sound is 3.065 seconds (see around Line 61 of GenreFeatureData.py). This is the default “self.timeseries\_length”, if you record shorter or longer sound files, you need to modify the programs to make it work which is a challenging task. See appendix for details.
   * Prepare the training data: Record each NAME for **at least 22 times**, so you need at least 22\*5 files in total. (Record and use more training samples for training to increase accuracy if you have time). Save them using the following format, for example:
     + Save files Peter.00000.au -- Peter.00009.au, etc. in \gtzan\\_train
     + Save files Peter.00010.au -- Peter.00014.au, etc. in \gtzan\\_test
     + Save files Peter.00015.au -- Peter.00019.au, etc. in \gtzan\\_validation
     + Save files Peter.00020.au -- Peter.00021.au, etc. in audio
     + Hints: Use GOLDWAV ( <https://www.goldwave.com/>) to record .au sounds, the default sample rate is 48000 , procedures in GLODWAV:
       1. Use "top-menu\_File/New" and select sampling rate 48000, or the function (icon) "record" to record many (22 or more) words in one go.

-- use any sampling you like (48000 is the best, but at least 22100) however, it is better to be consistent to all recordings. However, LIBROSA resample the load files using 22020 samples/second again, so use 22020 throughout the calculation. See <https://librosa.org/blog/2019/07/17/resample-on-load/#:~:text=The%20sampling%20rate%20%2D%2D%2D%20typically,in%20the%20discretely%20sampled%20signal>

* + - 1. After recorded the 22 words of one type (10 for training, 5 for test, 5 for validation, and 2 for prediction test) in one go, use the mouse to select individual words one by one.
      2. For each word use "top-menu/edit/copy to/use sun(.au) format." to save one word to one file, e.g. 'Peter.00000.au', 'Peter.00001.au' etc.
  + Rename the classes in your program.
    - Our approach is similar to the Music genre classification program. The genre here in your assignment is the name prefixed to the .au sound file. E.g., Peter.
    - To achieve that you have to edit and change the file GenreFeatureData.py
    - Around Line 10 of GenreFeatureData.py, the system will use these names as the class\_names. E.g. Change “Classical” 🡪 “Peter”

=======from line 10 of GenreFeatureData.py

hop\_length = None

genre\_list = [

"classical",

"country",

"disco",

"hiphop",

"jazz",

"metal",

"pop",

"reggae",

]

In this document, I am using the old function names as in music genre classification and don’t change it for the speech recognition system. To make less change to the original system can help you to run the system successfully. You can use suitable names in your program to reflect their true functions better (But I would recommend you change it after you successfully run your first system). The general description of the Music genre classification demo program can be found below:

* + - * <http://www.cse.cuhk.edu.hk/~khwong/www2/cmsc5707/Tutorial_LSTM_music_genre_classification.docx>
      * [hong'slinks - 1.Tensor\_windows (installation through Anaconda3) (google.com)](https://sites.google.com/site/hongslinks/1-tensor_windows-installation-through-anaconda3?authuser=0)
      * <https://github.com/ruohoruotsi/LSTM-Music-Genre-Classification>
  + Now if your data and system is ready, run the Music genre classification
  + **Step 1** – start tensorflow) assume you use conda with tensorflow, see section (4.b) above for tool installation
    - In anaconda prompt, Change Directory cd to the directory containing lstm\_genre\_classifier\_keras.py
    - conda>> activate tf-cpu , or activate tf-gpu # if you have a gpu
  + **Step 2** – build the model and weights) In the current directory, delete stm\_genre\_classifier\_lstm.h5 and all .npy files under \gtzan, they may come from previous training tests
    - Edit the file lstm\_genre\_classifier\_keras.py
    - Add the following line at the end of the file lstm\_genre\_classifier\_keras.py

# serialize model to JSON, this is to reflect the modle you create

model\_json = model.to\_json()

with open(“model.json”, “w”) as json\_file:

json\_file.write(model\_json)

# # serialize weights to HDF5

# model.save\_weights(“model.h5”)

print(“Saved model model\_json to disk”)

* + - Run the classification code: (tf-cpu)>> python lstm\_genre\_classifier\_keras.py
    - You will obtain the accuracy and loss parameters, save and present them in your report.
    - The model “model.json” and weight file (lstm\_genre\_classifier\_lstm.h5) are created,
      * copy model.json to \weights\model.json” (overwrite if necessary).
      * rename lstm\_genre\_classifier\_lstm.h5 to become model\_weights.h5, and copy model\_weights.h5 to \weights\ model\_weights.h5 (overwrite if necessary)
  + **Step 3** – classify a new sound file: Use prediction\_example.py to classify a new sound (e.g. Peter.00020.au) using the following instruction:
    - (tf-cpu)>> python predict\_example.py \audio\Peter.00020.au,
      * If successfully classified, you will see: Model predict: Peter
    - If the above steps are successful, you can modify the code lstm\_genre\_classifier\_keras.py by adding more layers, change parameters etc. to increase the accuracy.
      * Hints: See <http://www.cse.cuhk.edu.hk/~khwong/www2/cmsc5707/Tutorial_LSTM_music_genre_classification.docx> for the structure of the music genre classification system.
      * In lstm\_genre\_classifier\_keras.py, when using model.add(), set return\_sequences=True if it is a hidden layer.
      * In GenreFeatureData.py, The original features extraction uses all MFCC 13 parameters included MFCC0, for speech signals, MFCC0 may not help, can remove it. Other feature may not be useful of speech processing as well.
      * data[i, :, 0:13] = mfcc.T[0:self.timeseries\_length, :]
      * data[i, :, 13:14] = spectral\_center.T[0:self.timeseries\_length, :]
      * data[i, :, 14:26] = chroma.T[0:self.timeseries\_length, :]
      * data[i, :, 26:33] = spectral\_contrast.T[0:self.timeseries\_length, :]

1. **What to submit**:
   * **Write a brief report (1-2 pages) of your system include information such as**
     + The hardware system you used, PC version, Windows or Linux, or IOS version
     + Discuss how you improve your approach based on the framework in original music genre classification framework, such as increase layers, change parameters etc.
     + Show and compare the results of the original and your modified system :
       - 1. Accuracy
         2. Run time
   * **Follow the submission procedure on page 1.**