Project Report
Speech Recognition

Junaid Ahmad Bhat

February 8, 2021

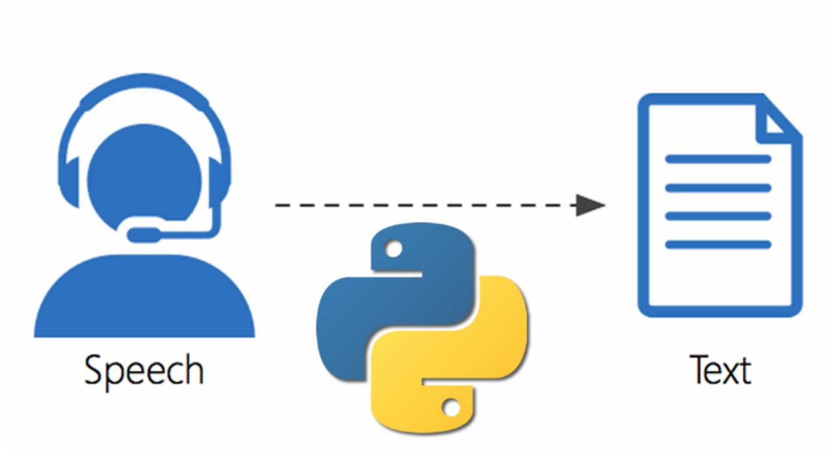


Figure 1:

Overview

Speech recognition, also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, is a capability which enables a program to process human speech into a written format. While it's commonly confused with voice recognition, speech recognition focuses on the translation of speech from a verbal format to a text one whereas voice recognition just seeks to identify an individual user's voice.

The most frequent applications of speech recognition within the enterprise include the use of speech recognition in mobile devices.Speech recognition can also be found in word processing applications like Microsoft Word, where users can dictate what they want to show up as text.For example, individuals can use this functionality in smartphones for call routing, speech-to-text processing, voice dialing and voice search. A smartphone user could use the speech recognition function to respond to a text without having to look down at their phone. Speech recognition on iPhones, for example, is tied to other functions, like the keyboard and Siri and Bixby on samsung android mobile phones also work on speech recognition. If a user adds a secondary language to their keyboard, they can then use the speech recognition functionality in the secondary language (as long as the secondary language is selected on the keyboard when activating voice recognition. To use other functions like Siri, the user would have to change the language settings.)

From smart lights that turn on or off on your command, Google Home Assistant that can place space trivia with you and make financial transactions when requested, Alexa that also works on same algorithm to follow your voice commands; speech recognition is the component of the system that makes it all possible.

Language used
python

Data

Record 80 utterances of each command.

Set 16KHz as sampling rate.

Save samples

Record/Forward

Record/Back

Record/LEFT

Record/Right

Record/STOP

Libraries used

Kapre,Scikit Learn,Tenserflow,Soundfile,Numpy,keras etc

Description

Using Convolutional layers ahead of LSTM is shown to improve performance in several research papers.

BatchNormalization layers are added to improve convergence rate.

Using Bidirectional LSTM is optimal when complete input is available. But this increases the runtime two-fold.

Final output sequence of LSTM layer is used to calculate importance of units in LSTM using a FC layer.

Then take the dot product of unit importance and output sequences of LSTM to get Attention scores of each time step.

Take the dot product of Attention scores and the output sequences of LSTM to get attention vector.

Add an additional FC Layer and then to output Layer with SoftMax Activation.

The model is successfully built and has achieved the accuracy of 99.78%(avg).

Code

→Requirements

1.kapre:Keras Audio Preprocessors - compute STFT, ISTFT,Melspectrogram, and others on GPU real-time.

2.Soundfile:SoundFile is an audio library based on libsndfile, CFFI and NumPy.SoundFile can read and write sound files. File reading/writing is supported through libsndfile, which is a free, cross-platform, open -source (LGPL) library for reading and writing many different sampled sound file formats that runs on many platforms including Windows, OS X, and Unix.

3.ffmpeg-python:There are tons of Python FFmpeg wrappers out there but they seem to lack complex filter support. ffmpeg-python works well for simple as well as complex signal graphs.FFmpeg is extremely powerful, but its command-line interface gets really complicated rather quickly - especially when working with signal graphs and doing anything more than trivial.
→ Then we import drive.

→Import section:

In this section we import different python libraries that we are required to use in the code as we proceed.

→Generate Data:

In this section we actually trace data into arrays and save it.

→Splitting Data into Train and Test:

In this section we split data into two groups: Training group and test group.

Test size has been has 20% and rest is for training purpose.

Also random state is set as integer instead of None,
the random state parameter is used for initializing the internal random number generator, which will decide the splitting of data into train and test indices in your case.

If random state is None or np.random, then a randomly-initialized Random State object is returned.
If random state is an integer, then it is used to seed a new RandomState object.
This is to check and validate the data when running the code multiple times. Setting random state a fixed value will guarantee that the same sequence of random numbers is generated each time you run the code.

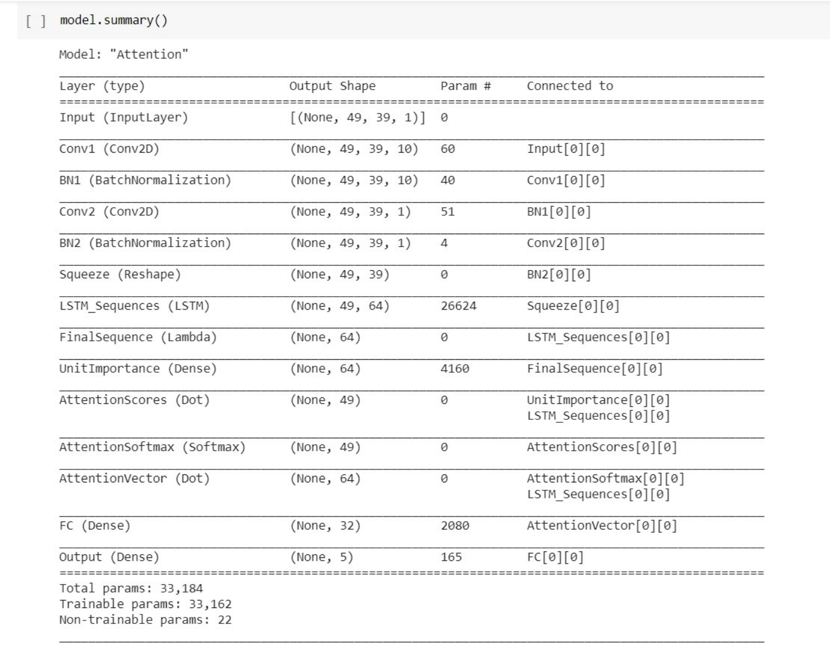
→Extrtacting Features:

This section all about scaling date to the level,so that the results are accurate upto large extent.

Melspectogram:The mel scale is a non-linear transformation of frequency scale based on the perception of pitches. The mel scale is calculated so that two pairs of frequencies separated by a delta in the mel scale are perceived by humans as being equidistant.In machine learning applica tions involving speech and audio, we typically want to represent the power spectrogram in the mel scale domain. We do that by applying a bank of overlapping triangular filters that compute the energy of the spectrum in each band.

Normalization:Normalization refers to rescaling real-valued numeric attributes into a 0 to 1 range. Normalization makes the features more consistent with each other, which allows the model to predict outputs more accurately.

→Model Summary:



```
[ ] model.summary()

Model: "Attention"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Input (InputLayer)              [(None, 49, 39, 1)]  0

__________________________________________________________________________________________________
Conv1 (Conv2D)                  (None, 49, 39, 10)   60          Input[0][0]
__________________________________________________________________________________________________
BN1 (BatchNormalization)        (None, 49, 39, 10)   40          Conv1[0][0]
__________________________________________________________________________________________________
Conv2 (Conv2D)                  (None, 49, 39, 1)    51          BN1[0][0]
__________________________________________________________________________________________________
BN2 (BatchNormalization)        (None, 49, 39, 1)    4           Conv2[0][0]
__________________________________________________________________________________________________
Squeeze (Reshape)               (None, 49, 39)       0           BN2[0][0]
__________________________________________________________________________________________________
LSTM_Sequences (LSTM)           (None, 49, 64)       26624       Squeeze[0][0]
__________________________________________________________________________________________________
FinalSequence (Lambda)          (None, 64)           0           LSTM_Sequences[0][0]
__________________________________________________________________________________________________
UnitImportance (Dense)          (None, 64)           4160        FinalSequence[0][0]
__________________________________________________________________________________________________
AttentionScores (Dot)           (None, 49)           0           UnitImportance[0][0]
                                                                 LSTM_Sequences[0][0]
__________________________________________________________________________________________________
AttentionSoftmax (Softmax)      (None, 49)           0           AttentionScores[0][0]
__________________________________________________________________________________________________
AttentionVector (Dot)           (None, 64)           0           AttentionSoftmax[0][0]
                                                                 LSTM_Sequences[0][0]
__________________________________________________________________________________________________
FC (Dense)                      (None, 32)           2080        AttentionVector[0][0]
__________________________________________________________________________________________________
Output (Dense)                  (None, 5)            165         FC[0][0]
==================================================================================================
Total params: 33,184
Trainable params: 33,162
Non-trainable params: 22
__________________________________________________________________________________________________
```

Figure 2: Model Summary

→Train Model:

In this section,we actually train our model with the data that we have already set for training purpose(80% in our case).

Verbose:verbose takes values as 0, 1, or 2. Verbosity mode. 0 = silent, 1 = progress bar, 2 = one line per epoch. Note that the progress bar is not particularly useful when logged to a file, so verbose=2 is recommended when not running interactively (eg, in a production environment).

Epochs: Integer. Number of epochs to train the model. An epoch is an iteration over the entire x and y data provided. Note that in conjunction with initial epoch, epochs is to be understood as ”final epoch”. The model is not trained for a number of iterations given by epochs, but merely until the epoch of index epochs is reached.

Batch size: Integer or None. Number of samples per gradient update. If unspecified, batch size will default to 32. Do not specify the batch size if your data is in the form of datasets, generators, or keras.utils.Sequence instances (since they generate batches).

Sparse Categorical Accuracy:Calculates how often predictions matches integer labels.

→Attention:

Attention is simply a vector, often the outputs of dense layer using softmax function. Before Attention mechanism, translation relies on reading a complete sentence and compress all information into a fixed-length vector, as you can image, a sentence with hundreds of words represented by several words will surely lead to information loss, inadequate translation, etc. However, attention partially fixes this problem. It allows machine translator to look over all the information the original sentence holds, then generate the proper word according to current word it works on and the context. It can even allow translator to zoom in or out (focus on local or global features). Attention is not mysterious or complex. It is just an interface formulated by parameters and delicate math. You could plug it anywhere you find it suitable, and potentially, the result may be enhanced.

Run

The Code is written in Google Colab.

Open ColabNotebook.ipynb and change Runtime to GPU

Upload **/content/drive/MyDrive/Record/** to Colab.

Run the cells in the same order as in Notebook.

Test

Tenserflow:TensorFlow is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.

Steps:

Locate the folder where you save your model.h5 file.

Start speaking when you see mike in the bottom right pane of the task

bar or see red blinking dot in the title bar.

If still facing some error try to set the mike recorder time in the previous

record function up to 3 sec.

Accuracy of the model can be increased by having large dataset i,e large no. of audio file for particular command and for all.