End to End Speech Recognition with Pykaldi/Deepspeech, webrtcvad, precise

Prajwal Rao

1 Introduction

This repository is an end to end speech recognition toolkit which uses open source libraries such as PyKaldi, Deepspeech , WebrtcVAD, mycroft precise

2 Installation

- Install Anaconda

```
conda create --name speech python=3.6
conda activate speech
```

- Install PyKaldi

```
conda install -c pykaldi pykaldi
```

- Install Deepspeech

```
pip3 install deepspeech # Install deepspeech-gpu if you want CUDA support
# Download the model
curl -LO https://github.com/mozilla/DeepSpeech/releases/download/v0.7.4/audio-0.7.4.tar.gz
tar xvf audio-0.7.4.tar.gz
```

- Install WebrtcVAD

```
pip install webrtcvad
```

- Install mycroft-precise

```
ARCH=x86_64
wget https://github.com/MycroftAI/precise-data/raw/dist/$ARCH/precise-engine.tar.gz
tar xvf precise-engine.tar.gz
sudo apt-get install portaudio19-dev
pip install pyaudio
pip install precise-runner
```

3 Steps

3.1 Download models

- Download Kaldi model from here
- Link all the hard coded paths in main.py (you might have to create online.conf, ivector$_{\text{extractor}}$.conf)
- Precise models have to be generated (data can be obtained here)

3.2 Train the WakeWord Detector using the following steps:

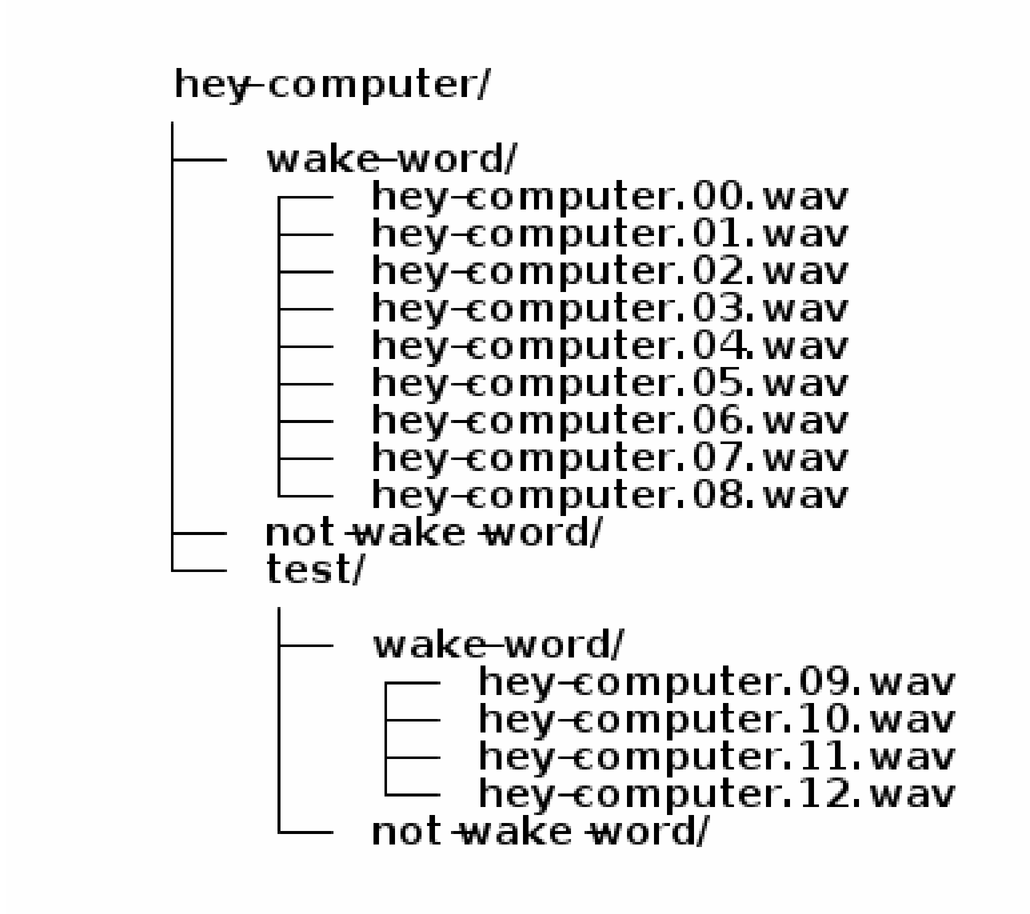
- Activate your anaconda environment
- Record your audio samples using

  ```
  precise-collect
  ```

- If you are recording by other means, convert the samples to 16kHz 1 channel 16-bit PCM wav audio files

  ```
  ffmpeg input.mp3 -acodec pcm_s16le -ar 16000 -ac 1 output.wav
  ```

- Make a folder sequence of this manner



```
hey-computer/
│
├──  wake-word/
│    ├── hey-computer.00.wav
│    ├── hey-computer.01.wav
│    ├── hey-computer.02.wav
│    ├── hey-computer.03.wav
│    ├── hey-computer.04.wav
│    ├── hey-computer.05.wav
│    ├── hey-computer.06.wav
│    ├── hey-computer.07.wav
│    └── hey-computer.08.wav
├──  not-wake-word/
└──  test/
     │
     ├──  wake-word/
     │    ├── hey-computer.09.wav
     │    ├── hey-computer.10.wav
     │    ├── hey-computer.11.wav
     │    └── hey-computer.12.wav
     └──  not-wake-word/
```

- Once the data is ready Train it for 60 epochs

  ```
  precise-train -e 60 hey-computer.net hey-computer/
  ```

- You can test your code using precise-test

  ```
  precise-test hey-computer.net hey-computer/
  ```

- The accuracy will be low and the false activation's will be high. To account for this we have to augment data with background

  ```
  mkdir -p data/random
  wget http://downloads.tuxfamily.org/pdsounds/pdsounds_march2009.7z
  7z x pdsounds_march2009.7z # Install p7zip if not yet installed
  cd ../../
  SOURCE_DIR=data/random/mp3
  DEST_DIR=data/random
  for i in $SOURCE_DIR/*.mp3;
  ```

```
do echo "Converting $i..."; fn=${i##*/};
   ffmpeg -i "$i" -acodec pcm_s16le -ar 16000 -ac 1 -f wav "$DEST_DIR/${fn%.*}.wav";
done
```

- Fine-tune your model with the augmented data

  ```
  precise-train-incremental hey-computer.net hey-computer/
  ```

- You can test the accuracy of your system using:

  ```
  precise-test hey-computer.net hey-computer/
  ```

- Convert your model to Tensorflow model

  ```
  precise-convert hey-computer.net
  ```

- To test your code in python use the sample$_{\text{precise}}$.py file, Change the model path to the required destination and run the code

  ```
  conda activate speech
  python sample_precise.py
  ```

3.3 Run the main code to test the pipeline

```
conda activate speech
python main.py
```

3.4 Using the API

- The simple way is to call the SpeechRecon as an Object and then use the run method
- The object consists of record variable which can be set to either True or False as per requirement

  ```
  from main import SpeechRecon
  speech_pipeline = SpeechRecon(record=False)
  speech_pipeline.run()
  ```

4 Results

5 Authors

- Prajwal Rao