Computing Assignment 3: Automatic Word Recognition Due: November 3, 2022

Submit your method description, code (python, html), testing data observations, results and discussion

The task: To recognize an uttered word by any speaker from a command word set.

Dataset: Subset of Google Speech Commands Dataset version 0.01

Audio files: 16 kHz sampling rate, each recording is of duration of 1 second or less (unpadded), stored as 16-bit PCM (signed integers).

The 10 words are: "down", "go", "left", "no", "off", "on", "right", "stop", "up", "yes"

Training set:
Each word has about 2100 recorded instances in the training set across several different speakers.
Thus, the total number of utterances in the training set is about 21,000.

There are approx. 1060 speakers in the training set and approx. 120 speakers in the test set for each word. There is no overlapping between the speakers in the training set and the test set.

Noise files - "doing_the_dishes.wav", "exercise_bike.wav", "white_noise.wav", "dude_miaowing.wav", "pink_noise.wav", "running_tap.wav" (each of approx. 1 minute)

You will use the training speech data as such or create new training data instances by adding noise to the available recordings depending on the test task at hand. Further, you can set aside a part of your training set (e.g. 20%) for validation (tuning any hyper-parameters of your models) if required.

Task A: Recognition of clean speech utterances

The test set contains around 250 recording instances for each word. These recordings have ambient noise but no further added noise. Thus the test set size in total is 2500 files.

Task B: Recognition of noisy utterances

The test set contains about 250 instances for each word. 10 dB SNR noise from any one of the above noises is added to each recording with probability 0.5. Thus about half of the samples in the test set have noise added to them. The test set size is about 2500.

Evaluation metrics – Accuracy (Percentage of words in the test set that are recognized correctly). Obtain the confusion matrix and consider how you can improve accuracy based on this.

Methods – Carry out speech end-pointing, pre-emphasis and MFCC feature extraction; next implement any one or more of the pattern matching methods studied so far: VQ codebook matching (bag of frames), template matching using DTW, or the statistical method of GMM-HMM (word level HMMs).

Note: DTW - Can be implemented from scratch (however, dtw python implementations are available)

GMM-HMM - Can be trained using the hmmlearn package

Dataset Details:
Folder Structure:
background_noise_ - Contains the noise files (listed above)
train - Contains 10 subfolders, one for each word. Each subfolder contains the training samples for that word (no overlapping with any of the test sets)
test_clean - Test set for Task A. Contains 10 subfolders, one for each word. Each subfolder contains the test samples for that word
test_noisy - Test set for Task B. Contains 10 subfolders, one for each word. Each subfolder contains the test samples for that word
README.txt - A README file
1804*.pdf: a paper describing the Google dataset

File naming convention for train, test_clean and test_noisy (files with no noise added) sets:

SID_nohash_**i**.wav
SID - Speaker ID
i - Indexing required to differentiate between same word samples spoken by the same speaker

File naming convention for test_noisy (files with noise added) set:

SID_nohash_**i**_n**x**.wav
SID - Speaker ID
i - Indexing required to differentiate between same word samples spoken by the same speaker
x - Integer to indicate which noise was added (0 - “doing_the_dishes.wav”, 1 - “dude_miaowing.wav”, 2 - “exercise_bike.wav”, 3 - “pink_noise.wav”, 4 - “running_tap.wav”, 5 - “white_noise.wav”)

Note: A python function for adding 10dB SNR noise has been provided.