university of groningen

university of groningen

Speech Recognition - I

Demonstrator

Isolated Word Recognition Engine

This demonstrator is based on HTK (Hidden Markov Model Toolkit) that is used for research and development work in automatic speech recognition. It has been developed by the Machine Intelligence Laboratory (formerly known as the Speech Vision Robotics Group) at the Cambridge University Engineering Department. It is widely used for research in speech processing. The source code of HTK should be downloaded from http://htk.eng.cam.ac.uk/. One needs to register at the website before downloading HTK source code. 'HTKbook' may also be downloaded from the same website for assistance, although the following instructions should be adequate to perform this experiment.

Initial Steps

Recording: This can be done by any audio editor cum recorder software like Audacity, Adobe Audition, Praat, Wavesurfer etc.

1. All the recordings should be done in 16-bit PCM at 16 KHz in .wav format.
2. Each student should record 3 instances of the spoken digits 0-9 in quiet, noise-free environment. All these digits should be stored in isolation in different files. **The silence preceding and following the word utterance should be removed manually from all the files before saving**.
3. The format of the filenames should be xxxx_y_z. Here xxxx refers to first four alphabets of your name, y refers to the digit recorded (0-9) and z refers to the recording instance (1-3) of that digit. A valid filename can be *abcd_9_2.wav*.
4. All the students must mail their zipped folder of thirty *.wav* files (3 instances of 10 digits) to all other students and course instructor **latest by 17th January 2022 midnight CET.** Each student should download all the other students' recordings which should be used for the training database.
5. **All the three instances of each digit from all students except your own recordings should be taken for creating the training database. One instance of each digit in one's own voice should be taken by each individual student for the respective testing.** Note that the training database consists of other students' .wav files while the testing database consists of one's own .wav files.
6. The training files should be stored in a folder TRAIN while testing files should be stored in a folder TEST.

Installing HTK: (The full procedure described here is for Linux only.)

1. You will have to register on the HTK website to download the HTK source code.
2. After downloading and untaring the source code go to the HTK folder and type -

   ./configure

university of groningen

make
make install

3. If “make” command gives errors related to X11 library, ./configure –disable Hslab should be tried for configuring instead of ./configure.

Training

The purpose of this section is to create 10 HMMs corresponding to the 10 digits.

Preparing file path lists, HMM initial definition scripts and config files: All these files should be created in the same folder e.g. 'experiment'. All the commands should also be executed from the same folder.

1. Prepare a list of ***absolute paths*** of ALL training files by the name 'training_wav.list.scp'. It may look like -

/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_1.wav
/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_2.wav
/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_3.wav
.
.
/home/name/Desktop/Demonstrator/TRAIN/Segments/five/abcd_5_1.wav
.
.
/home/name/Desktop/Demonstrator/TRAIN/Segments/ten/abcd_9_3.wav
etc.

2. Prepare a similar file list called training_mfclist.scp. This file will contain target paths of the MFCC files. The only difference with the previous file is that '.wav' extension should be replaced by '.mfc'.

3. Prepare a list of training MFCC files of each single digit. The file should be named training_0_mfc.scp for digit 0, training_1_mfc.scp for digit 1 and so on. The file containing .mfc files for digit 1 looks like -

/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_1.mfc
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_2.mfc
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_3.mfc
.
.
.
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_2.mfc
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_3.mfc

4. A file training.scp should be created. This file can be created using the command: *paste -d ' ' training_wavlist.scp training_mfclist.scp >training.scp*. The first column of this file

university of groningen

'training.scp' will contain .wav file paths of all the training files. The second column will contain MFCC file paths (MFCC files are not created yet). This file will look like:

/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_1.wav
/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_1.mfc
/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_2.wav
/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_2.mfc
/home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_3.wav
home/name/Desktop/Demonstrator/TRAIN/Segments/zero/abcd_0_3.mfc
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_1.wav
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_1_1.mfc
.
.
/home/name/Desktop/Demonstrator/TRAIN/Segments/nine/abcd_9_3.wav
/home/name/Desktop/Demonstrator/TRAIN/Segments/one/abcd_9_3.mfc
etc.

This file will be used later for creating MFCC files from .wav files. The MFCC file in the 2nd column in the above file will create MFCC vectors corresponding to the .wav file in the first column.

5. Make a folder HMM0. In this folder, keep the initial HMM definitions with the names ZERO, ONE, TWO, THREE, FOUR, FIVE, SIX, SEVEN, EIGHT and NINE. **Keep the filenames as instructed in the previous sentence as these files will be used for testing**. The HMM definition file ZERO can be copied from below:

```
~o <VecSize> 26 <MFCC_0_D>
~h "ZERO"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 26
     0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 26
     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
  <State> 3
    <Mean> 26
     0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 ten 0.0
    <Variance> 26
     1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 0.0
  <State> 4
    <Mean> 26
```

university of groningen

```
    0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0
   <Variance> 26
    1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0
 <TransP> 5
  0.0 1.0 0.0 0.0 0.0
  0.0 0.6 0.4 0.0 0.0
  0.0 0.0 0.6 0.4 0.0
  0.0 0.0 0.0 0.7 0.3
  0.0 0.0 0.0 0.0 0.0
<EndHMM>
```

Carefully look at the definition of the HMM. One may note that this is a 5-state HMM with one mixture in each state. The header of this HMM states that this HMM should be created from 26 component feature vectors. The string *MFCC_0_D* means MFCC vectors should be appended with an MFCC energy component and 1st order difference coefficients.

The definitions of HMMs of all the other digits are exactly same except the "~h" option of the second line. Argument of "~h" must be changed to the digit name i.e. ZERO, ONE, TWO etc.

6. Create scripts for initializing HMMs and then re-estimating HMMs:

Script name: hinit_script.sh
Contents:

```
HInit -T 2 -S train_0_mfc.scp -H HMM0/ZERO -M HMM1  ZERO
HInit -T 2 -S train_1_mfc.scp  -H HMM0/ONE -M HMM1  ONE
HInit -T 2 -S train_2_mfc.scp  -H HMM0/TWO -M HMM1  TWO
HInit -T 2 -S train_3_mfc.scp  -H HMM0/THREE -M HMM1  THREE
HInit -T 2 -S train_4_mfc.scp  -H HMM0/FOUR -M HMM1  FOUR
HInit -T 2 -S train_5_mfc.scp  -H HMM0/FIVE -M HMM1  FIVE
HInit -T 2 -S train_6_mfc.scp  -H HMM0/SIX -M HMM1  SIX
HInit -T 2 -S train_7_mfc.scp  -H HMM0/SEVEN -M HMM1  SEVEN
HInit -T 2 -S train_8_mfc.scp  -H HMM0/EIGHT -M HMM1  EIGHT
HInit -T 2 -S train_9_mfc.scp  -H HMM0/NINE -M HMM1  NINE
```

Script name: hrest_script.sh
Contents :

```
HRest -T 2 -S train_0_mfc.scp    -H HMM1/TEN -M HMM2  ZERO
HRest -T 2 -S train_1_mfc.scp    -H HMM1/ONE -M HMM2  ONE
HRest -T 2 -S train_2_mfc.scp    -H HMM1/TWO -M HMM2  TWO
HRest -T 2 -S train_3_mfc.scp    -H HMM1/THREE -M HMM2  THREE
HRest -T 2 -S train_4_mfc.scp    -H HMM1/FOUR -M HMM2  FOUR
HRest -T 2 -S train_5_mfc.scp    -H HMM1/FIVE -M HMM2  FIVE
HRest -T 2 -S train_6_mfc.scp    -H HMM1/SIX -M HMM2  SIX
HRest -T 2 -S train_7_mfc.scp    -H HMM1/SEVEN -M HMM2  SEVEN
HRest -T 2 -S train_8_mfc.scp    -H HMM1/EIGHT -M HMM2  EIGHT
```

university of groningen

HRest -T 2 -S train_9_mfc.scp -H HMM1/NINE -M HMM2 NINE

7. A config file is needed to define the characteristics of MFCCs which are to be created from .wav files. Create 'config' file as -

```
TARGETKIND = MFCC_0_D
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
SOURCEFORMAT=WAV
```

Once these lists, definitions, scripts and config file are created, the following three commands are needed to create the 10 HMMs corresponding to the 10 digits.

1. *HCopy -A -D -T 1 -C config -S training.scp*: This command creates MFCCs (2nd column of training.scp) from source wav files (1st columns of training.scp). The parameters for MFCC creation are defined in config file.
2. *sh hinit_script.sh*: This script takes initial definitions of 10 HMMs defined in HMM0 directory and initializes parameters of 10 HMMs and puts them in HMM1. For this task it picks up MFCCs from respective MFCC files which are provided in different lists given by arguments of -S.

3. *sh hrest_script.sh* : This script takes initial definitions of 10 HMMs as found in HMM1 directory and re-estimates parameters of 10 HMMs and puts them into folder HMM2. For this task it picks up MFCCs from the respective mfcc files which are provided in different lists given by arguments of -S.

The HMM2 directory now contains the finally trained HMMs.

Testing

1. MFCC Generation – This can be done for the test .wav files as before in the training step. The list file test.scp has to be created which contains the paths of .wav files in the first column and paths of .mfc files in the second column. The command used for MFCC creation is *HCopy -A -D -T 1 -C config -S test.scp*.
2. Grammar creation - gram.txt should be created with following contents:

```
$WORD = ZERO | ONE | TWO | THREE | FOUR | FIVE | SIX | SEVEN | EIGHT | NINE;
(($WORD)
```

university of groningen

(This means that the test file will contain only one word from the ten words listed above.)

3. slf creation - HVite does not accept the grammar file directly. .slf file is created using the command - *HParse -T 1 gram.txt net.slf*

4. Dictionary - This file should be created to map words to HMM names. The first column is words, the last one is HMM names, the middle one is the name to be listed in the recognition result. The contents of this file should look like –

ZERO [ZERO] ZERO

ONE [ONE] ONE

TWO [TWO] TWO

THREE [THREE] THREE

FOUR [FOUR] FOUR

FIVE [FIVE] FIVE

SIX [SIX] SIX

SEVEN [SEVEN] SEVEN

EIGHT [EIGHT] EIGHT

NINE [NINE] NINE

5. Testing slf file created in step 3 (optional) - This can be done using the command : *HSGen -n 10 net.slf dict.txt* .This would result in any 10 words from our list.

6. Viterbi decoding of test segments - We need to give the ten HMMs, slf file, the test mfcc file list, list of HMM names. A list of HMM names *hmmlist.txt* and *test_mfclist.scp* should be created.
Hmmlist.txt will look like:

ZERO
ONE
TWO
THREE
FOUR
FIVE
SIX
SEVEN
EIGHT
NINE

The file test_mfclist.mfc will contain a list of paths of MFCC test files. The final command for

university of groningen

decoding of MFCC test files is given below. Final result will be indicated in reco.mlf.

HVite -H HMM2/ZERO –H HMM2/ONE -H HMM2/TWO -H HMM2/THREE -H HMM2/FOUR
-H HMM2/FIVE -H HMM2/SIX -H HMM2/SEVEN -H HMM2/EIGHT -H HMM2/NINE -i reco.mlf -w net.slf dict.txt hmmlist.txt -S test_mfclist.scp

Final Submission

Submit the following for the purpose of evaluation:
1. HMM2 folder containing the final HMMs of 10 digits.
2. A report having-
   A. Two HMM definition files THREE and FIVE.
   B. reco.mlf obtained after the last step with your observations on the method and the test performance.

Appendix

The meanings of some of the options used with HTK scripts are:
-A: causes current command line arguments to be printed
-D: displays current configuration settings
-T int: To input trace levels
-C file: To input configuration file
-M dir: To indicate the directory for output HMM definitions.
-H: To specify H input HMM definition files.
-S file: To specify the list of input mfcc or wav or both kind of files.