Phone recognition with HTK toolkit

==========

This document is a tutorial for phone recognition using the HTK toolkit.
The objective of the tutorial is to support the students of EE619 to learn how to use the HTK toolkit and perform phone recognition on the TIMIT corpus.
You can purchase the TIMIT corpus from below.
http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1
This document provides the detailed procedures of phone recognition from scratch, from downloading the toolkit to computing the phone-recognition accuracy.
This provides all terminal commands and scripts you need for each procedure.
They were tested on 'aludra' machine (with Solaris), an account of which is provided to all EE619 students.
Although this document assumes that you run it on Solaris, you can run on linux, too. (It was tested on Ubuntu 12.04 with 64-bit CPU.)

This document is written by Jangwon Kim.
Signal Analysis and Interpretation Laboratory (SAIL)
University of Southern California
17. Jan. 2013
- Updated in April 2 2015

If you have any question, please email to Jangwon: jangwon@usc.edu

Special thanks to Naveen Kumar, Kartik Audhkhasi, Prof Panayiotis G. Georgiou and Prof. Shrikanth S. Narayanan for advice and help.

==========

Note:
- Almost all commands you need to type are provided in bold characters.
- Be familiar with any text editor, e.g., vi, vim, emacs and pico, for Solaris. The best way is to play with it for a few minutes and read some online tutorials.
- Final results reported in this tutorial can be different from yours. The results in this tutorial are based on too small training data due to the space limit for each student account in aludra.

[HTK setup]

(0) Access to aludra with your account.
ssh (your email id)@aludra.usc.edu
ex) jangwon@aludra.usc.edu
password is same to your USC email password
- change your shell from tcsh to bash by following
bash

(1) Download HMM-toolkit (HTK-3.4.1.tar.gz) and HTK-Sample.tar.gz (this file is just for testing your installation) to your own computer (not aludra).
- Firstly, you need to register for downloading HTK toolkit (HTK source code (tar+gzip archive), HTK samples (tar+gzip archive)). Go to "http://htk.eng.cam.ac.uk/" and do it.

(2) Move the downloaded files (HTK-3.4.1.tar.gz) to aludra.
You can use scp, filezilla, etc. You can download filezilla from USC.
e.g.) For Ubuntu: Go to the directory of the two files and type the following:
scp HTK-3.4.1.tar.gz HTK-Sample.tar.gz jangwon@aludra.usc.edu:~

From here, you are going to work on only in aludra.
(3) Unpack HTK-3.4.1.tar.gz in aludra
- Access to aludra using ssh (read (0) above)
- Check whether you have both **HTK-3.4.1.tar.gz** and **HTK-Sample.tar.gz.**

(4) compile HTK (read README file CAREFULLY)
- Unpack
gunzip HTK-3.4.1.tar.gz
tar xvf HTK-3.4.1.tar
cd htk

- You should read this part in README:
Solaris: if "make" isn't installed you may need to add /opt/sfw/bin and /usr/ccs/bin to your path and run "./configure MAKE=gmake" with any other options you require. Then run "gmake" instead of "make", alternatively you can create a symbolic link called "make" somewhere it your path to /opt/sfw/bin/gmake

- My comment on it: Don't worry too much about above instruction. Even though you don't understand, it is fine. Just move on to the following.

- Add path, configure, gmake
export PATH=$PATH:/opt/sfw/bin:/usr/ccs/bin
./configure MAKE=gmake
gmake

- Add path for the directory HTKTools
e.g.) export PATH=$PATH:/home/scf-11/jangwon/htk/HTKTools

- Note: Adding path using “export” is temporary. So whenever you log in to aludra, you should export the directory first, then continue. DON’T FORGET ABOUT IT.

[TEST your HTK compile]

(1) Unpack HTK sample
- Unpack
cd ..
gunzip HTK-samples-3.4.1.tar.gz
tar xvf HTK-samples-3.4.1.tar
cd samples
cd HTKDemo

- Now, You SHOULD read README in the directory HTKDemo.

(2) Generate directories where output files of each process will do.
mkdir -p accs hmms hmms/hmm.0 hmms/hmm.1 hmms/hmm.2 hmms/hmm.3 hmms/tmp proto test

(3) Run the demo
./runDemo configs/monPlainM1S3.dcf

- If you can see “HTK Results Analysis” box, then

- Compare your results in the box “HTK Results Analysis” with the results in the README file. They may not match exactly, but they should be very close. If they are close, you are done with HTK setup.

[Get TIMIT database]

- Download partial TIMIT database from the below website. You can simply type below address in your web browser. Password will be given by email upon your request.
http://sail.usc.edu/~jangwon/EE619materials/timit_EE619_dr1.tar.gz
(This is a subset of the TIMIT corpus, which will be provided to the students of EE619

temporally only for education purpose. For the entire TIMIT corpus, see http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC93S1)

- Move the timit_EE619_dr1.tar.gz to the home directory of your aludra account. You can use filezilla, scp and so forth.
e.g.) scp timit_EE619_dr1.tar.gz jangwon@aludra.usc.edu:~/

- Unpack the dataset as follows:
gunzip timit_EE619_dr1.tar.gz
tar xvf timit_EE619_dr1.tar

[Phone Recognition in batch]

(1) Download directory asr_batch which contains all scripts, lists, etc.
- Download it from the following address to your machine, then copy it to aludra.
http://sail.usc.edu/~jangwon/EE619materials/asr_batch.tar.gz (for EE619 students only)
Or, http://sail.usc.edu/~jangwon/asr_batch_release.tar.gz (for public)

- Change all paths in doall_comments.py (wrapper for all processes) in asr_batch/configs/timit/htk/ee619_scripts/ accordingly.

- Change paths in asr_batch/lists/timit/*.ctl files accordingly. Do it only for those in doall_comments.py. Don't have to fix the other *.ctl files.

(2) Run doall_comments.py
cd ./asr_batch/configs/timit/htk/ee619_scripts/
python doall_comments.py

Final results will be similar to the followings:
```
====================== HTK Results Analysis =======================
  Date: 'Today'
  Ref : >gwon/workspace/ee619spring2013/asr_batch/benchmark/timit/htk/test.mlf
  Rec : >rkspace/ee619spring2013/asr_batch/benchmark/timit/htk/rec_mono_m2.out
---------------------- Overall Results ----------------------
SENT: %Correct=0.00 [H=0, S=88, N=88]
WORD: %Corr=46.33, Acc=37.58 [H=1330, D=452, S=1089, I=251, N=2871]
====================================================================
```

[Phone Recognition step-by-step]

This section is the main in this tutorial.
This section provides specific commands, explanations, and dependencies (necessary files and their formats) for the whole phone recognition process.
The order of the process follows the order of the script doall.py mostly.
It is highly recommended to read the HTK book
You MUST read Chapter 3 in the HTK book at least.
The example in the Chapter 3 is very similar to what you will do in this section, so it will be very useful.
The order of process in this tutorial is a little bit different from the Chapter 3 in HTK book.

(0) Preparation
- Set up (creating a working directory and export path of HTKTools)
cd ~/
mkdir phoneRecogDetailed
cd phoneRecogDetailed
export HTKToolsBin="/home/scf-11/jangwon/htk/HTKTools";

Note: Check whether you exported the path well by following command.
ls $HTKToolsBin

"export" is a function to set up *temporary* path, so you should run "export ~" command whenever you open your terminal.

- Download templates and scripts from below address. Then, unpack it.
http://sail.usc.edu/~jangwon/EE619materials/scripts_and_templates.tar.gz (for EE619 students only)
Or, http://sail.usc.edu/~jangwon/scripts_and_templates_release.tar.gz (for public)
gunzip scripts_and_templates.tar.gz
tar xvf scripts_and_templates.tar

(1) MFCC feature extraction
- Feature extraction using HCopy requires two files as inputs: a config file that has coding parameters, and a script file that has pairs of wav file paths and corresponding mfcc feature file paths.

-- Create the list of train files (audio files) and the list of test files. You should include path of the audio files. Following commands assume that you have timit corpus in the home directory of your account. ex) /home/scf-11/jangwon in my case.
cd ~/
find ~/timit/train/ -iname *.wav > ./phoneRecogDetailed/train.wavlist
find ~/timit/test/ -iname *.wav > ./phoneRecogDetailed/test.wavlist

-- Create the list of feature files (mfcc files) of train set and the list of feature files of test set. You should include path.

cd ~/phoneRecogDetailed/
mkdir -p features/train/ features/test/
cat train.wavlist | sed -e "s/\/home\/scf-11\/jangwon\///g" -e "s/\//_/g" -e "s/wav/mat/g" -e "s/timit/\/home\/scf-11\/jangwon\/phoneRecogDetailed\/features\/train\/timit/g" > train.matlist
cat test.wavlist | sed -e "s/\/home\/scf-11\/jangwon\///g" -e "s/\//_/g" -e "s/wav/mat/g" -e "s/timit/\/home\/scf-11\/jangwon\/phoneRecogDetailed\/features\/test\/timit/g" > test.matlist

-- Paste the two lists (the list of audio files and the list of mat files) into a script file for each of train and test data.

paste train.wavlist train.matlist > train.list
paste test.wavlist test.matlist > test.list

-- For a config file, you can use config.parming in directory “Templates”.
Please check what configuration is used for MFCC extraction.

-- Extract MFCC features using HCopy

$HTKToolsBin/HCopy -T 1 -C ./Templates/config.parming -S train.list
$HTKToolsBin/HCopy -T 1 -C ./Templates/config.parming -S test.list

(2) Create wordnet

Wordnet is a file that is written in a low level notation, called HTK Standard Lattice Format (SLF).
This file includes the information of each class (phone in this tutorial) instance and class-to-class transition. HTK toolkit provides a converter, HParse that automatically creates a user-readable file (grammar) to SLF file (wordnet).
For grammar, you should write in a specific way supported by HTK toolkit. For more details about it, please read chapter 3.1.1 in HTK book. It is straightforward, so please read it.
Since your task is phone recognition, you need a grammar for just a phone sequence.
It is trivial to create a grammar file from the list of phone, so please try to create your grammar file, and compare the one provided in the Templates directory.

- Create a wordnet file from a grammar file, using HParse as following.

$HTKToolsBin/HParse grammar wdnet

(This command assumes that the file name of your grammar file is “grammar.”)

(3) Create dictionary

You need a dictionary that defines the transition map between the lexical unit of your transcript and corresponding phone sequences. If your task were word recognition, you may

need to generate a dictionary mapping all words of the corpus you are using and their phone sequences (, or just use an popularly used dictionary, i.e., CMU dictionary). However, for phone recognition, you can simply use a dummy phone dictionary with a phone transcript. The dummy phone dictionary should be written as following.

AA AA
AE AE
AH AH
AO AO
…
Y Y
Z Z
ZH ZH
sil sil

Make sure that you have only “sil” for all silence notations (sp, sil, etc.) in your transcript. This is not common, but this tutorial does that for the sake of simplicity.
It is also very trivial to generate this dummy phone dictionary, so create yours and compare with the one provided in Templates directory.

(4) Create Master Label File (MLF) for the phone transcripts of all files.
For training of a phone recognizer, we need a MLF file.
HTK toolkit uses MLF files or compactness.
A MLF file has information about what file corresponds to what phone (or word).
In this tutorial, phone transcriptions should be provided in the form of a Master Label File (MLF) for training.
For the details of MLF file format, please read pages 28 - 29 of the HTK book.
This tutorial provides phone transcripts (timit_train_dr1.phone.ref, timit_test_dr1.phone.ref. see Templates directory) for all feature files for the subset of TIMIT (only DR1). Create your MLF file. This tutorial provides a python script for the task (see scripts directory). Go to scripts directory and change the paths in map2mlf.py accordingly. Then, do the following.
python map2mlf.py
Now, you are ready to train acoustic models!

(5) Define monophone HMM structure (prototype)
- Define a prototype model for monophone HMM. Parameter values do not matter in this prototype file. It just defines the model topology. For the details of prototype format, read chapter 3.2.1 in HTK book. The present tutorial provides a prototype file (see `proto’ in Templates directory). ‘Proto’ has left-to-right configuration with 3 states and 39 dimensional feature vectors. Open the file and see what it is.

(6) Initialization of the prototype model
Use HCompV to initialize the prototype model with means and variances from the training

data.
- Create a directory for initial model
cd .. (Note: it assumes that you were in scripts directory)
mkdir -p models/hmm0
- Initialization with HCompV with randomly chosen 500 feature files in train set (500prem.matl includes the list of the feature files). Please fix the paths of files in 500prem.matl accordingly.
$HTKToolsBin/HCompV -C Templates/config.parmed -f 0.01 -m -S ./Templates/500prem.matl -M models/hmm0 Templates/proto

- Using generated “proto” and “vFloors” files, and create a master macro file, “hmmdefs.”
The file ‘hmmdefs’ contains the copies of the "proto" HMM renamed to each of the required models (the HMM models of all phones, including sil’s). Why does HTK Toolkit use this file? Use of this file allows you to avoid creating many HMM files (like, one model file for each phone). See Fig. 3.7 in page 33 in HTKbook for the format of hmmdefs file. This tutorial provides a python script to generate hmmdefs automatically. See ‘gen_hmmdefs.py’ in scripts directory. You should change paths in gen_hmmdefs.py accordingly before you run it.
python ./scripts/gen_hmmdefs.py
Check what hmmdefs is look like. Does it have HMM model topology for each phone?

(7) Tune HMM parameters (model training)
Use HERest to re-estimate the parameters of the HMM of each phone by forward-backward algorithm.
- Create a directory for models such that the number of Gaussian mixture is 1.
mkdir -p models/hmm1
- Copy initial model file (the one that you created in step (6)) to directory hmm1.
cp models/hmm0/hmmdefs models/hmm1/

- Re-estimation of HMM parameters using training set
$HTKToolsBin/HERest -A -C Templates/config.parmed -S train.matlist -H models/hmm0/vFloors -H models/hmm1/hmmdefs -I train.mlf -M models/hmm1 -t 250.0 150.0 1000.0 Templates/local_phone_list
- Repeat this re-estimation process 3 times or more (usually continue this process until log-likelihood is saturated, meaning your model does not fit much better to the train data. But, let’s try just 5 times, which is sufficient for this partial dataset).

(8) Initialization of the prototype model with N number of mixtures, N > 1
You can skip this section (8) if you want a single Gaussian instead of mixtures of Gaussian.
Use HHEd to modify your hmm definition file for N number of mixture case.
- create split.hed file and write below text in it.
  For example if you want the number of mixture to be 2, then.
MU 2 {*.state[2-4].mix}

The file “split.hed” is also included in template for N=2 case (see Templates directory) so that

you can compare with yours.
- Create new hmmdefs file for the number of mixture = N case, using HHEd and split.hed.
mkdir -p models/hmm2
cp models/hmm1/* models/hmm2/
$HTKToolsBin/HHEd -H models/hmm1/hmmdefs -M models/hmm2 Templates/split.hed Templates/local_phone_list
(Ignore if you see WARNING [-2637].)

Then repeat section (7 - re-estimation process) from hmm2

(9) Viterbi decoding
Perform viterbi decoding using HVite as follows.
$HTKToolsBin/HVite -A -C ./Templates/config.parmed -w wdnet -H models/hmm1/hmmdefs -S test.matlist -i recog.out -p -10 -s 1.0 ./Templates/local_dic ./Templates/local_phone_list

(10) Evaluation
You can evaluate your models using HResults at the sentence level and word level. See pp 289 of the HTK book for details.
$HTKToolsBin/HResults -A -I test.mlf -e ??? sil ./Templates/local_phone_list recog.out

Result will be similar to the following:
```
====================== HTK Results Analysis =======================
  Date: 'Today'
  Ref : test.mlf
  Rec : recog.out
------------------------ Overall Results --------------------------
SENT: %Correct=0.00 [H=0, S=88, N=88]
WORD: %Corr=47.09, Acc=38.59 [H=1352, D=465, S=1054, I=244, N=2871]
============================================================
```