Speech Processing (Module CS5241)

Semester 2, AY2012/2013

Assignment 4: Spoken Letter Recognition

1 Files Provided

The following files are provided to you for **Assignment 4**.

- assignment4/ # Root directory
  - data/ # Data directory
    - wav/
      - train/ # Directory containing training data feature files
      - test/ # Directory containing test data feature files
  - lib/ # Library directory
    - cfgs/
      - mfcc.cfg # Configuration file for MFCC feature type
      - plp.cfg # Configuration file for PLP feature type
    - dicts/
      - letters.dct # Dictionary using letters as acoustic units
    - lists/
      - letters.list # Model list file for all letters
      - words.list # A list of English words
    - labs/
      - letters.mlf # Master label file using letters as units
      - letters_sil.mlf # Same as above plus start/end sil
    - protos/
      - proto.mfcc # A prototype file with MFCC features
      - proto.plp # A prototype file with PLP features
    - nets/
      - letters.loop.net # A simple letter network
      - letters.unigram.net # A letter unigram network
      - letters.bigram.net # A letter bigram network

2 Data Preparations

The name of the WAV files have the following format:

`<userid>_<wordid>.wav`

where each user is given a unique `<userid>` (*e.g.* `user01`, `user02`, ...). The WAV file contains the spoken letter sequence which corresponds to the spelling of the word indicated by `<wordid>`.

2.1 Preparing File Lists

Unlike Assignment 3, you need to extract the appropriate acoustic features from the WAV file. HTK supports the following feature types: LPC, MFCC and PLP. You can use `HCopy` to convert the WAV files into feature files. For example, to convert to MFCC features, you issue the following command:

```
$ HCopy -C lib/cfgs/mfcc.cfg -S lib/flists/wav2mfcc.scp
```

The HTK *script file* (`lib/flists/wav2mfcc.scp`) will look something like this:

```
data/wav/train/user01_BELIEVER.wav data/mfcc/train/user01_BELIEVER.mfcc
data/wav/train/user01_DISPERSE.wav data/mfcc/train/user01_DISPERSE.mfcc
data/wav/train/user01_ILLINOIS.wav data/mfcc/train/user01_ILLINOIS.mfcc
data/wav/train/user01_PACKAGES.wav data/mfcc/train/user01_PACKAGES.mfcc
...
```

The first column of the script file contains the source WAV files. The second column contains the target feature file. You need to make sure that the directories for the target feature files already exist. Once you created the feature files, you need to create the train/test script files for each feature type (to be used for training and testing). For example, you might want to create the following files for the MFCC features:

```
lib/flists/train.mfcc.scp
lib/flists/test.mfcc.scp
```

You can use `HList` to examine the content of the feature files.

2.2 Vocabulary and Lexicon Preparation

Before training the acoustic models, the vocabulary and lexicon (pronunciation dictionary) need to be prepared. For this assignment, two sets of dictionary and model list files are provided depending on the choice of acoustic units. If the letters are chosen as the acoustic units, you can use the following dictionary and model list file:

`lib/lists/letters.list`
`lib/dicts/letters.dct`

2.3 Transcription File Preparation

A transcription file is needed for training the HMMs and evaluating system outputs. For this assignment, the letter transcription files are given in the HTK's Master Label File (MLF) format:

`lib/labs/letters.mlf`
`lib/labs/letters_sil.mlf`

The second file contains additional 'sil' labels at the start and end of each utterance. One of these files should be used as transcriptions for training while the other should be used as reference transcription for performance evaluation. Do you know which is for what?

3 Acoustic Model Training

Please follow the HMM training steps in the `readme.pdf` document for **Assignment 3** to build the acoustic models.

4 Experiments

Below are the list of items to do for Assignment 4:

1. **Building the basic models**

   You need to make use of the instructions given in Section 2 and Section 3 to train and evaluate the acoustic models. Train ~~three~~ two sets of acoustic models, one for each of the following feature types:

   - MFCC
   - PLP

   For the basic systems, letters are used as the acoustic units. One HMM is used to represent one letter. The HMMs have **three emitting states** with a **left-to-right** topology. Perform Baum-Welch (BW) training with iterative mixture splitting to increase the number of Gaussian components per state in the following order: $1 \rightarrow 2 \rightarrow 4 \rightarrow 6 \rightarrow 8$. Perform 4 BW training iterations after each mixing up.

   Evaluate the performance of `hmm14`, `hmm24`, `hmm44`, `hmm64` and `hmm84` using the simple letter network (`lib/nets/letters.net`).

2. **Varying the number of states**

   Three emitting states per HMM may not be optimum for your system. Train different sets of models with different number of states and evaluate their performance. Typically, there is a compromise between the number of states and the number of Gaussian components per state. Report the best three configurations that you have tried.

3. **Adding differential parameters**

   Previously, you have used only 12 dimensional features (`MFCC` or `PLP`) to train your acoustic models. Consider adding energy terms and differential parameters to your acoustic features. This can be achieved by changing the TARGETKIND setting in the configuration file. The table below indicates the qualifiers that can be appended to the basic features:

Qualifier	Meaning
`_E`	energy
`_0`	zeroth coefficient
`_D`	first differential
`_A`	second differential
`_T`	third differential
`_Z`	per file mean normalization

   Therefore, the following configuration

```
TARGETKIND = MFCC_E_D_A
```

yields 39-dimensional features. For different features, a new set of HMMs need to be trained.

4. **Using decoding networks with n-gram probabilities**

   So far, you have only used the simple letter network to perform recognition. In this assignment, you are also provided with two additional decoding networks:

   - `letters.unigram.net` – a decoding network with unigram probabilities
   - `letters.bigram.net` – a decoding network with bigram probabilities

   Choose the best system you have so far and perform recognition using the above decoding networks. What do you observe?

5. **Using word-constrained decoding networks**

   If the spoken letter sequences were based on *known* English words, then a custom decoding network can be constructed to constrain the decoding search space to include only letter sequences which correspond to the known words.

   Supposed that the known words are given by the word list in `lib/lists/words.list`, construct a custom decoding network to take advantage of this new piece of information. (*Hint: You can use* `HParse` *to construct custom networks by writing grammar rules*)

6. **Bonus Questions:**

   (a) **Speaker Adaptation**

   Refer to the **HTK Book** for details on how to perform unsupervised speaker adaptation. Are you able to improve the basic models using speaker adaptation?

   (b) **Feature Projection**

   Refer to the **HTK Book** for details on how to perform Heteroscedastic Linear Discriminant Analysis (HLDA). Are you able to improve the basic models using HLDA feature projection?

Dr. Sim Khe Chai
February 2013