# PASCAL CHiME Challenge: baseline recogniser and scoring scripts

**Ning Ma (n.ma@dcs.shef.ac.uk), Jon Barker and Heidi Christensen, October 2010 (version 0.1)**

## 1 Introduction

This distribution contains a 'standard' HMM-based speech recogniser and a scoring tool for use in the PASCAL CHiME Challenge (http://www.dcs.shef.ac.uk/spandh/chime/challenge.html). Using this package you should be able to:

- obtain keyword recognition scores from formatted result files (*do_score_all.sh*);
- perform recognition and score the challenge data (*do_recog_all.sh*);
- estimate parameters of speaker dependent HMMs (*do_train_all.sh*).

Detailed usage of these scripts will be given in Section 3. Note that the estimated HMMs are already included in this package but the relevant scripts are still supplied for reference. The use of the recogniser is also optional and is designed for participants whose algorithms produce 'separated' or 'enhanced' waveforms from the noisy signals. The package automates the entire process of producing recognition results. For those of you who have your own recogniser, the package will automate and more importantly **standardise** the process of obtaining keyword recognition scores on the challenge data.

While extensive testing has been performed, we would appreciate rapid feedback on any problems that you find. Please report problems to Ning Ma (n.ma@dcs.shef.ac.uk).

## 2 Contents of the distribution

The root directory of this package contains a README, 8 subdirectories and 3 shell scripts:

| | |
|---|---|
| **bin/** | HTK binaries compiled for 32-bit Linux machines (i686) |
| **do_recog_all.sh** | extract MFCC features, recognise and score |
| **do_score_all.sh** | generate keyword recognition score |
| **do_train_all.sh** | estimate speaker dependent models |
| **etc/** | config files for the recogniser |
| **forced_alignments/** | automatically generated forced alignments |
| **flists/** | file lists for various data sets |
| **labels/** | master label files for the training set |
| **models/** | HMM definitions |
| **README.pdf** | this document |
| **results/** | recognition results |
| **scripts/** | auxiliary programs |

## 3 Usage

All the examples given here assume the current working directory is the package root directory. The HTK package is needed to run the recognition and training programs. This distribution already includes HTK binaries compiled for 32-bit Linux machines (architecture=i686), which should work on most Linux machines. If you find the HTK binaries are not working for you, you may want to contact us for support, or obtain your own copy of HTK by visiting http://htk.eng.cam.ac.uk/download.shtml.

## 3.1 Scoring script: do_score_all.sh

Even if you have your own recogniser, you should use the scoring script **do_score_all.sh** to produce keyword recognition accuracy. For each condition, your algorithm should produce a results file whose name is of the form PREFIX_SNR.txt, where PREFIX can be anything (e.g. 'baseline_mfcc_devel') and SNR is one of the SNR subconditions for the dataset {m6dB, m3dB, 0dB, etc ...}. For example, you might use the name baseline_mfcc_devel_0dB.txt for the results of your development set 0 dB condition. Put all the results files in the same directory (e.g. 'results/devel.mfcc'). You should use the same prefix (here, 'baseline_mfcc_devel') for all of your results files which are in the same directory so that the scoring script finds them all.

Each result file should contain one line per utterance. The first item on the line is the name of the file being processed, e.g. s1_bgaa9a. The remainder of the line should contain the keywords part of the recognition result: letter and digit. An example result file might be

```
s1_bgaa9a a 9
s1_bgwi1a y 1
...
```

**The challenge is scored over only the two keywords: letter and digit**. The scoring procedure is as follows: each utterance receives a score of 0, 1 or 2, depending on how many of the (letter, digit) keywords are correct. The score reported is the average of the scores across utterances, expressed as a percentage.

Once your score files are prepared, you can produce a score summary by running the script like this:

```
./do_score_all.sh RESULT_DIR
```

where RESULT_DIR is the directory containing the results files (e.g. 'results/devel.mfcc').

If you want to score a single result file, you can use the script **do_score.sh** in the 'scripts' subdirecotry:

```
./scripts/do_score.sh baseline_mfcc_devel_0dB.txt
```

## 3.2 Recogniser script: do_recog_all.sh

If your algorithms produce 'enhanced' waveforms or MFCC features from the noisy signals, you can use the supplied recogniser to produce recognition results. You should ensure that your separated waveforms or MFCC features are held in a directory structure identical to the original mixed waveforms, i.e. the directory should have subdirectories for the various SNRs. The recogniser will only report results for the SNR conditions you provide in the directory. To use the recogniser, simply run:

```
./do_recog_all.sh SETNAME DATAROOT
```

where SETNAME is used as a unique ID for the data set stored in DATAROOT and DATAROOT is supposed to contain subdirectories {m6dB, m3dB, 0dB, etc ...}. The recognition script goes through each of the SNR conditions that it finds in DATAROOT and performs the following steps: i) convert the waveforms into MFCC features; ii) runs the recogniser, and iii) scores the results. If a SNR subdirectory contains MFCC features instead of waveforms, they will be used directly by the recogniser.

For example, suppose your save the original isolated development set in 'PCCdata16kHz/devel/isolated'. To obtain the baseline recognition results use the command

```
./do_recog_all.sh orig_devel PCCdata16kHz/devel/isolated
```

The result files will be saved in the directory 'results/orig_devel.mfcc' and you should obtain precisely the following keyword recognition accuracies:

| -6 dB | -3 dB | 0 dB | 3 dB | 6 dB | 9 dB |
|---|---|---|---|---|---|
| 31.08 | 36.75 | 49.08 | 64.00 | 73.83 | 83.08 |

These are the baseline 'do nothing' results.

If your processed waveforms of the development set are in a directory called 'processed_devel', then simply run:

```
./do_recog_all.sh devel2 processed_devel
```

The result files will be saved in the directory 'results/devel2.mfcc'.

By default the extracted MFCC acoustic feature files are saved in the directory 'features' within the root directory of this package. To avoid writing feature files into this directory you could either make a symbolic link named as 'features' to somewhere else, or modify the variable '$FEAT_ROOT' in do_recog_all.sh.

## 3.3 Training script: do_train_all.sh

Although estimated HMMs are supplied along with the package, the script used to obtain the HMMs is also included. You should be able to obtain identical HMM definitions by using:

```
./do_train_all.sh WAVROOT
```

where WAVROOT is the directory that holds training waveforms in each of the subdirectories {id1, id2,... id34} for each speaker accordingly. For example, suppose you download the CHiME Challenge training set and save it in 'PCCdata16kHz/train/reverberated'. Simply run

```
./do_train_all.sh PCCdata16kHz/train/reverberated
```

The trained HMM definitions will be saved in 'models/mfcc' in the package root directory.

By default the extracted MFCC acoustic feature files are saved in the directory 'features' within the package root directory. To avoid writing feature files into this directory you could either make a symbolic link named as 'features' to somewhere else, or modify the variable '$FEAT_ROOT' in do_train_all.sh.

If you compute MFCC acoustic features separately and they are in a directory (e.g. features/train) that has the identical directory structure (i.e. features for each speaker are saved in a separate subdirectory such as id1, id2... of the directory), then you can use 'do_train.sh' in the 'scripts' subdirectory directly:

```
./scripts/do_train.sh mfcc features/train
```

where 'mfcc' tells the script the acoustic features in 'features/train' are MFCC features.

## 4 Recogniser details

The baseline recogniser is based on the HTK package version 3.4.1.

The raw speech waveforms are parameterised into standard 39-dimensional Mel Frequency Cepstral Coefficients (MFCCs) applied with Cepstral Mean Normalisation (CMN), i.e. 12 Mel-cepstral coefficients and the logarithmic frame energy plus the corresponding delta and acceleration coefficients and then applying CMN (MFCC_E_Z_D_A).

The words are modelled as whole-word HMMs with a left-to-right model topology with no skips over states and 7 Gaussian mixtures per state with diagonal covariance matrices. The number of states for each word is based on 2 states per phoneme:

| | |
|---|---|
| **4 states:** | at by in a b c d e f g h i j k l m n o p q r s t u v x y z one two three eight |
| **6 states:** | bin lay place set blue green red white with four five six nine now please soon |
| **8 states:** | again zero |
| **10 states:** | seven |

Speaker dependent HMMs are employed. They are estimated by using a set of trained speaker independent HMMs as the starting point, and performing 4 more iterations of EM training using the 500 training utterances for each speaker. The HMMs are trained using reverberant signals without any noise and there is no retraining on noisy signals.

The following grammar is used during recognition:

$\$verb = bin|lay|place|set;$
$\$colour = blue|green|red|white;$
$\$prep = at|by|in|with;$
$\$letter = a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|x|y|z;$
$\$digit = zero|one|two|three|four|five|six|seven|eight|nine;$
$\$coda = again|now|please|soon;$

$(\$verb\ \$colour\ \$prep\ \$letter\ \$digit\ \$coda)$

The CHiME corpus provides binaural signals, but the baseline recogniser employs monaural signals by averaging the two channels together in waveforms.