

The TRANSCRIBER BOOK

Valtcho Valtchev

Version 1.0

Dan Kershaw
Julian Odell

The Transcriber Book (for Transcriber Version 1.0)

©COPYRIGHT 1998 Entropic Cambridge Research Laboratory

All Rights Reserved

Contents

I	Truetalk Transcriber Overview	1
1	Introducing Truetalk Transcriber	3
1.1	Package components	3
1.1.1	Acoustic models	3
1.1.2	Language models	3
1.1.3	Pronunciation dictionaries	3
1.1.4	The HAPI interface	4
1.1.5	Transcriber tools	4
1.2	System development life cycle	4
1.2.1	Collecting adaptation data	5
1.2.2	System evaluation	5
1.2.3	Prototyping and implementation	5
2	Transcriber Tutorial	7
2.1	Speaker/environment adaptation	7
2.1.1	Configuring the recogniser	7
2.1.2	Data collection	7
2.1.3	Using the recogniser	8
2.2	Creating an LVX system for a specific vocabulary	9
2.3	The “Sherlock Holmes” recogniser	12
3	Advanced adaptation techniques	15
3.1	Supervised adaptation	15
3.2	Unsupervised adaptation	16
3.3	Testing the new TMF and MMF	17
3.4	Adaptation types	17
3.5	Choosing an appropriate model set	19
3.6	Guide for incremental adaptation	19
3.6.1	Incremental class-based adaptation	19
3.6.2	Incremental full adaptation	19
3.7	Initialising models for adaptation	20
II	Transcriber tools	21
4	System evaluation tool	23
4.1	A tour of the main window	23
4.1.1	The menu bar	23

4.1.2	The Controls	25
4.1.3	The Output Panel	26
4.1.4	The Current Panel	26
4.2	Configuring the recogniser	26
4.2.1	System configuration files	26
4.2.2	Choosing a system configuration	27
4.3	Loading the recogniser	29
4.4	Source settings and calibration	29
4.5	Performing recognition	31
4.6	N-best results and playback	32
4.7	Speaker/environment adaptation	33
4.7.1	Supervised and unsupervised adaptation	33
4.7.2	Notes	34
5	Enrollment tool	37
5.1	A tour of the main window	37
5.1.1	The menu bar	37
5.1.2	The waveform panel	39
5.1.3	The Prompt area	39
5.1.4	The Controls	39
5.1.5	The Status area	40
5.2	Collecting adaptation data	40
5.3	Resuming enrollment	43
6	HTK and HAPI based tools	45
6.1	HAPIAdapt	46
6.1.1	Function	46
6.1.2	Use	46
6.1.3	Tracing	47
6.2	HAPIVite	48
6.2.1	Function	48
6.2.2	Use	48
6.3	HCopy	52
6.3.1	Function	52
6.3.2	Use	52
6.3.3	Trace Output	54
6.4	HHed	55
6.4.1	Function	55
6.4.2	Use	64
6.4.3	Tracing	65
6.5	HList	66
6.5.1	Function	66
6.5.2	Use	66
6.5.3	Tracing	67
6.6	HResults	68
6.6.1	Function	68
6.6.2	Use	69
6.6.3	Tracing	71
6.7	HVite	72
6.7.1	Function	72

6.7.2	Use	72
6.7.3	Tracing	75
III	Language Modelling	77
7	Introduction	79
7.1	Introduction	79
7.2	N-Gram language models	79
7.3	Perplexity	80
7.4	LM tools in TRANSCRIBER	80
8	The Operating Environment	81
8.1	The Command Line	81
8.2	Script Files	82
8.3	Configuration Files	83
8.4	Standard Options	84
8.5	Error Reporting	84
8.6	Words and Strings	85
8.7	LM file formats	85
8.8	Summary	85
9	Language modelling tools	87
9.1	HLMCopy	88
9.1.1	Function	88
9.1.2	Use	88
9.1.3	Tracing	89
9.2	LAdapt	90
9.2.1	Function	90
9.2.2	Use	90
9.2.3	Trace Output	91
9.3	LPlex	92
9.3.1	Function	92
9.3.2	Use	92
9.3.3	Trace Output	93
10	Error and Warning Codes	95
10.1	Generic Errors	95
10.2	Summary of Errors by Tool and Module	96
IV	Appendices	99
A	Acoustic models	101
A.1	Telephony model sets	102
B	Language models	103
B.1	Mixed-case language model	103

C	Dictionaries	105
C.1	Output Symbols	105
C.2	Phone Sets	105
C.3	Mixed-case dictionaries	106

Part I

Truetalk Transcriber Overview

Chapter 1

Introducing Truetalk Transcriber

TRUETALK TRANSCRIBER¹ is a developers' toolkit for building large vocabulary continuous speech recognition (LVR) systems. The toolkit includes all components required for an LVR task - acoustic models, language models, dictionaries and a version of the HTK application programming interface (HAPI) incorporating a large vocabulary recognition decoder. In addition, TRANSCRIBER provides interactive visual tools for system evaluation and collection of speaker adaptation data.

1.1 Package components

1.1.1 Acoustic models

Phone-level acoustic Hidden Markov models are provided for each supported language (UK-English, US-English and UX-English for mixed UK/US applications). The model sets have been trained on clean-speech data from a wide range of speakers recorded using a close-talking microphone in low-noise conditions. An optional pack (**telpack**) is also provided which includes a US model set for telephony speech applications.

1.1.2 Language models

A 200,000 word statistical trigram language model is provided, trained from large corpora (approx. 0.5 billion words) of text data from the Business News domain. An optional pack (**lmpack**) is also provided which includes a mixed-case 200,000 word language model derived from approximately 1 billion words of text data from the North American business/general news domain.

1.1.3 Pronunciation dictionaries

A general purpose pronunciation dictionary is provided for each of the supported languages. For English language systems the dictionaries cover approximately 200,000

¹In the remaining parts of this book the name "TRANSCRIBER" will be used exclusively to refer to the TRUETALK TRANSCRIBER package.

words. The optional language pack (lmpack) provides mixed-case versions of the core English dictionaries.

1.1.4 The HAPI interface

TRANSCRIBER includes the HTK application programming interface (HAPI) which allows the application developer to incorporate speech capabilities into target applications. A JAVA version of the HAPI interface (JHAPI) is also included. The HAPI library available in TRANSCRIBER incorporates a speaker independent large vocabulary continuous speech decoder capable of recognising up to 65,000 word pronunciations. The decoder has been optimised to be fast, yet efficient in its memory usage. The latest release (1.3) of HAPI offers full support for static and dynamic adaptation, improved confidence scoring algorithms and reduced overall memory footprint. For a complete description of HAPI/JHAPI refer to the accompanying documentation, the HAPI Book.

1.1.5 Transcriber tools

The tools in the TRANSCRIBER package enable the developer to customize the supplied resources to a particular task domain. In particular, the supplied language models can be tailored to a specific size and vocabulary, updated to include new words, adapted to a new application domain using relevant text data (legal, medical, etc.) and evaluated on texts from that domain.

In speaker dependent and/or multiple speaker environments the supplied acoustic model sets can be adapted to a particular speaker yielding improved recognition accuracy and speed.

For existing HTK toolkit customers, the Transcriber package includes replacement versions of some the tools which will allow them to integrate models built with HTK into TRANSCRIBER.

Transcriber provides an interactive graphical user interface (GUI) based system evaluation tool. The tool allows the user to rapidly evaluate the performance of new systems. Example recognition systems with vocabularies of 6,000 and 60,000 words for British and American English are also included.

1.2 System development life cycle

In order to design a large vocabulary recognition system the user needs to create the following components: a word list defining the target system's vocabulary, a dictionary of word pronunciations, a domain-specific language model and an acoustic model set. Typically, the system development life cycle begins by defining the system's vocabulary using expert knowledge of the target domain. To assist the developer in this task, the TRANSCRIBER package includes word frequency lists which can be used to decide which words are to be included. Once a word list is formed, the developer can create the system's dictionary and language model from the supplied resources using tools in the TRANSCRIBER distribution. For new words (not already present in the supplied dictionaries) the developer is required to supply phonetic pronunciations and likely probabilities of occurrence. In most cases, an acoustic model set can be chosen directly from the supplied resources. All components of a speaker independent system are now ready. The newly created system can now be evaluated using the system

evaluation tool in TRANSCRIBER. In addition, speaker adaptation can be used to optimize the performance of the system for a particular talker using the enrollment tool.

1.2.1 Collecting adaptation data

The enrollment tool in TRANSCRIBER enables the user to record speech data for speaker adaptation purposes by reading out a sequence of utterance prompts. The recorded data is then used to compute a speaker/environment adaptation transform. Adaptation transforms are computed automatically by the enrollment tool at the end of each enrollment session. Alternatively, the HAPIADAPT tool in the TRANSCRIBER distribution can be used to compute self-contained speaker dependent model sets.

1.2.2 System evaluation

The GUI-based system evaluation tool in TRANSCRIBER allows the developer to evaluate the performance of a newly created LVR recognition system. Recognition can be performed from either pre-recorded waveform speech files or direct audio input using the machine's built-in audio hardware. For each recognised speech utterance results are displayed on the screen. To facilitate further scoring of the recognition output, the results can be optionally saved as lattices of multiple utterance hypotheses and/or an HTK-style master label files (MLFs). The system evaluation tool can also load and apply speaker adaptation transforms prior to performing recognition trials. On-line speaker adaptation is supported in both supervised and unsupervised modes.

The LVR recogniser and speaker adaptation facilities provided by the system evaluation tool are implemented using calls to the HAPI library. Thus, the system evaluation tool provides a realistic simulation of the recognition capabilities in the final user application.

1.2.3 Prototyping and implementation

Once the recogniser components have been created and tested, the speech recognition system can be incorporated into the user application. As noted earlier, TRANSCRIBER includes HAPI which enables the developer to create recogniser instances within the application, perform recognition of speech from a variety of input sources and pass results back to the application to trigger actions.

Developers should note that several run-time licensing options for porting Entropic's proprietary state of the art recognition technology to commercial products (via HAPI) are available. In addition, a special runtime version of HAPI has been developed, with very compact runtime requirements and improved recognition speed. Further details are available from Entropic.

Chapter 2

Transcriber Tutorial

This chapter presents three examples of using the TRANSCRIBER package to construct and evaluate large vocabulary recognition systems. The first example demonstrates the use of the speaker/environment adaptation facility to adapt the supplied UK model set. The second example shows how to create an LVX system for a specific task vocabulary. In the third example, the language model adaptation tool is used to adapt the supplied language model using text data from a new task domain.

2.1 Speaker/environment adaptation

This example demonstrates how to adapt one of the supplied systems to current environment/speaker characteristics.

2.1.1 Configuring the recogniser

Start the *System evaluation tool* by changing to the TRANSCRIBER distribution directory and executing the `lvrdemo` batch file. The tool's main window will appear on the screen. Choose **Configure Recogniser** from the **File** ↓ menu. The recogniser configuration dialog will appear on the screen. Choose the 60k UK system¹ and click **OK**. The acoustic models and the dictionary will be loaded.

2.1.2 Data collection

In order to adapt the chosen system we will need to collect speech data representative of the current user and environment. To record speech data, choose the **New enrollment** option from the **Adaptation** ↓ menu. This will bring up the *Enrollment tool* and the session configuration dialog. Fill-in the requested details such as name, gender, age, location, microphone, etc. This will allow you to accurately identify the generated adaptation transform in later trials. Change the session ID to a meaningful identifier, for example “djk01”. Click **OK**. This will create a session directory `djk01` in the specified location (default location is `sessions`) and load the default prompt file `prompts/red-headed.pmt`. The prompt file contains around 280 utterances, however

¹American English speakers should select the 60k US or UX system.

in this example we shall record only the first 50 utterances². Note that the collected waveform data will be stored in the `djk01/waveforms` directory and it can be added to and/or used later to adapt other systems.

Once the prompts have been loaded, the first utterance prompt will appear in the text area. Before recording data, calibrate the speech/silence detector by choosing **Calibrate source** from the **Settings ↓** menu. Click **Calibrate** and read out the prompt. The following measured values will be displayed:

- **Background level** - This is the background silence level. When the silence detector is being used, the silence level should be between 15 and 30dB. If the speech detector is disabled, **n/a** is displayed.
- **Mean speech level** - The mean speech level should be 30 to 40dB higher than the measured silence level to maintain recogniser accuracy (around 55 to 70dB). If the speech detector is disabled, **n/a** is displayed.
- **Dynamic range** - This measurement reflects the dynamic range of speech signal. The value should be less than 75% to ensure that the signal is not clipped. If the speech detector is disabled, **n/a** is displayed.

Click **Done** when finished. The system is now ready for recording. Click on **Record** and read out the prompt. If the prompt was recorded successfully the waveform data will be shown in the *Waveform panel*, an initial model adaptation will be performed and the next prompt will be displayed. The recording procedure can be repeated as necessary.

After recording 50 utterances terminate the enrollment procedure by clicking **Stop**. Click **Finish** to calculate an adaptation transform for the acoustic model set in the chosen recognition system. The session information will be saved in the specified sessions directory and a general adaptation transform will be computed from the collected speech data. At this point the user may choose to save the transform and exit. Alternatively, a refined transform can be computed by adapting the acoustic model and calculating a new transform from the data. Click **Refine and save** and wait for the second adaptation pass to complete.

2.1.3 Using the recogniser

The next step in this example is to load and evaluate the performance of the recogniser. Select **Load Recogniser** from the **File ↓** menu to commence the loading procedure. Once all components have been loaded the message “Loading done” will appear in the status panel. In this example we will configure the recogniser to run in speech/silence detection mode. Choose **Source settings** from the **Session ↓** menu. Select the desired source type (**Mic-in** or **Line-in**) and ensure that the speech/silence detector is switched on. Click **OK** to dismiss the dialog.

To perform recognition, click **Start**, wait for the message “Recognising from live audio” to appear in the *Status* panel and then speak. The speech/silence detector will automatically determine the end of the utterance once you have stopped speaking. The current partial recognition hypothesis will be displayed in the *Current* panel. Once recognition has finished, the recognised text will be displayed in the *Output*

²In general, the more data collected the better the transform is likely to be - i.e. improved recognition speed and accuracy

panel and the total utterance and recognition times will be shown in the *Current* panel. Click **Replay** to play back the recorded utterance. It is also possible to select a portion of the recognised text and replay the matching speech. Furthermore, clicking **N-Best** will display a list of possible alternatives for the currently selected text or for the whole utterance.

To adapt the acoustic models using the generated transform, choose **Load transform** from the **Adaptation** ↓ menu. A file selection dialog will appear. Navigate to the directory `sessions/djk01`, select `djk01.tmf` and click **Open** to load the adaptation transform. To display information about the loaded transform select **Transform info** from the **Adaptation** ↓ menu. A dialog box will appear listing the transform details. Click **OK** to dismiss.

Perform further recognition trials using the adapted model - the overall recognition performance should be better in terms of both accuracy and speed.

If you wish to revert to the original speaker independent model, choose **Unload transform** from the **Adaptation** ↓ menu. A dialog will appear asking you to confirm the unloading of the transform. Click **OK**. The adaptation transform will be unloaded and the acoustic models will be transformed back to their original state.

2.2 Creating an LVX system for a specific vocabulary

This example demonstrates the creation of the components of an LVX system for a specific vocabulary.

1. **Create vocabulary word list** - The vocabulary list should include all words in the task including the sentence start (`<s>`) and sentence end (`</s>`) markers. The `examples` directory of the TRANSCRIBER distribution contains the following example word list (`numbers.wlist`)

!COLON	!COMMA	!DASH
!DOUBLE-QUOTE	!ELLIPSIS	!EXCLAMATION-POINT
!HYPHEN	!LEFT-BRACE	!LEFT-PAREN
!PERIOD	!QUESTION-MARK	!RIGHT-BRACE
!RIGHT-PAREN	!SEMI-COLON	!SINGLE-QUOTE
</s>	<s>	A
AND	APRIL	AUGUST
BILLION	CENTS	DECEMBER
DEGREE	DEGREES	DOLLAR
DOLLARS	EIGHT	EIGHTEEN
EIGHTEENTH	EIGHTH	EIGHTHS
EIGHTY	ELEVEN	ELEVENTH
FEBRUARY	FIFTEEN	FIFTEENTH
FIFTH	FIFTHS	FIFTY
FIRST	FIVE	FORTY
FOUR	FOURTEEN	FOURTEENTH
FOURTH	GRADIAN	GRADIANS
HUNDRED	JANUARY	JULY
JUNE	MARCH	MAY

MINUTE	MINUTES	MILLION
NINE	NINETEEN	NINETEENTH
NINETY	NINTH	NINTHS
NOVEMBER	OCTOBER	OH
ONE	PENCE	PERCENT
POINT	POUND	POUNDS
QUARTER	QUARTERS	RADIAN
RADIANS	SECOND	SECONDS
SEPTEMBER	SEVEN	SEVENTEEN
SEVENTEENTH	SEVENTH	SEVENTHS
SEVENTY	SIX	SIXTEEN
SIXTEENTH	SIXTH	SIXTHS
SIXTY	TEN	TENTH
TENTHS	THE	THIRD
THIRDS	THIRTEEN	THIRTEENTH
THIRTIETH	THIRTY	THOUSAND
THREE	TWELFTH	TWELFTHS
TWELVE	TWENTIETH	TWENTY
TWO	ZERO	

Note that the TRANSCRIBER distribution includes a word frequency list for the most commonly encountered 200,000 words in Business News articles used to train the language models. This list is available in `resources/ECRL_200K.freq`.

2. **Verify coverage by supplied dictionaries.** Check to see if the task word list contains any words which are not in the supplied 200K word dictionaries (the words covered by the 200K are listed in `resources/ECRL_200K.wlist`). In the above word list, the following four words do not occur in the supplied dictionary - GRADIAN, GRADIANS, RADIAN, RADIANS.
3. **Create supplementary dictionary.** Create a supplementary dictionary file for the words not present in the supplied dictionary. Note that the pronunciations should be expressed using phones from the model set for the chosen language. The phone sets in the UK, US and UX English systems are available in appendix C. In our example, the supplementary dictionary for UK English contains the following pronunciations

```
GRADIAN  g r ay d iy ax n
GRADIANS g r ay d iy ax n z
RADIAN   r ay d iy ax n
RADIANS  r ay d iy ax n
```

and is available in `examples/numbers_uk_extra.dct`.

4. **Create supplementary language model.** Words not present in the supplied dictionary should also be added to the language model for the task. At present, the language model can be updated only by adding unigram entries for the previously unseen words. For our example, the supplementary unigram entries are stored in `examples/numbers.uni` and is listed below. Note that all unigram probabilities must be given as \log_{10} numbers.


```
-3.0 GRADIAN
-3.0 GRADIANS
-3.0 RADIAN
-3.0 RADIANS
```

To facilitate the creation of supplementary unigram files, the TRANSCRIBER distribution contains the unigram probabilities for the 200K words in the supplied dictionaries and language models. This list resides in `resources/ECRL_200K.uni`.

5. **Create task-specific dictionary and language model.** This procedure uses the HLMCopy tool to generate a task-specific dictionary and language model from the supplied and supplementary dictionaries and LMs. For the UK system, the command line is as follows

```
HLMCopy -T 1 -C prune.cfg -d numbers_uk_extra.dct \
-d ../resources/ECRL_UK_200K.dct -f ultra -u numbers.uni \
-v numbers_uk.dct -w numbers.wlist \
../resources/ECRL_BN1_200K.lm numbers.lm
```

The resulting dictionary and LM are available in `models/numbers_uk.dct` and `models/numbers.lm` respectively.

6. **Create system configuration file.** With all recognition components in place, the next step is to create a HAPI system configuration file. For the UK English, the configuration file (`config/uk_numbers.config`) is given below

```
## UK numbers
## British english models, numbers vocabulary
##

# Default is no tracing - can use suggested values to track problems

HAPI:    TRACELEVEL = 0  # 07777
HAudio:  TRACE = 0      #    3
HParm:   TRACE = 0      #   022
HNet:    TRACE = 0      #    1
HLM:     TRACE = 0      #    1

# models and other resources

HAPI:    MMF          = resources/EPCX2_UK01.mmf
HAPI:    HMMLIST      = resources/EPCX2_UK01.list
HAPI:    DICTFILE     = models/numbers_uk.dct
HAPI:    NETFILE      = models/numbers.lm

#include "hapilvx.cfg"
#include "coding16epc.cfg"
#include "adapt_uk.cfg"
#include "audio.cfg"
```

With the HAPI config file in place, the system evaluation tool can be invoked and the performance of the system tested.

2.3 The “Sherlock Holmes” recogniser

This example demonstrates the use of the LADAPT command line tool to create a target domain language model by adapting the supplied business news domain model. For this tutorial we have supplied a number of pre-processed Sherlock Holmes stories in the `examples/holmes` directory. All stories except one have been placed in the `train` directory and will be used to adapt the supplied language model `ECRL_BN1_200K.lm`. The “Red-headed league” story resides in the `test` directory and it will be used to evaluate the perplexity (see section 7.3) of the language models. Note that the following commands should be executed from the `examples/holmes` directory of the TRANSCRIBER distribution and the `bin.$CPU` directory should be in the execution path.

1. **Scan the “Sherlock Holmes” texts and obtain a word list.** This step relies on the *Perl* script `getwlist.pl` available in the `examples/holmes` directory.

```
perl ./getwlist.pl train/*.txt > holmes_data.wlist
```

The text data contains 18081 distinct words including the sentence start (`<s>`) and sentence end (`</s>`) symbols.

2. **Filter common words.** From the new list select all words which are present in the supplied 200K dictionary. This step relies on the *Perl* script `commwords.pl` available in the `examples/holmes` directory.

```
perl ./commwords.pl holmes_data.wlist \
  ../../resources/ECRL_200K.wlist > holmes_dict.wlist
```

The new word list contains 16312 words.

3. **Create target system’s vocabulary.** To widen coverage, first select the top 10K most common words from the supplied 200K list.

```
head -10000 ../../resources/ECRL_200K.freq | gawk "{print $1;}" \
  > 10k.wlist
```

Next, merge the newly generated 10K list with the words occurring in the Sherlock Holmes text data.

```
cat 10k.wlist holmes_dict.wlist | sort | uniq > holmes.wlist
```

The target system’s vocabulary contains 20091 words.

4. **Create dictionary and language model.** The language model is produced by pruning the supplied 200K business news LM to the `holmes.wlist` vocabulary and medium size (threshold of 15 for bigrams, 40 for trigrams). At the same time, HLMCOPY will also generate the target system’s dictionary from the supplied UK 200K dictionary.

```
HLMCopy -C ../prune.cfg -T 1 -c 2 15 -c 3 40 \
-d ../../resources/ECRL_UK_200K.dct -v models/holmes_uk.dct \
-w holmes.wlist ../../resources/ECRL_BN1_200K.lm \
models/holmes_BN1_med.lm
```

The resulting language model (`holmes_BN1_med.lm`) contains 1.6m bigrams and 1.2m trigrams. Note that this is still a business news domain model adjusted in size and vocabulary.

5. **Adapt the model.** Here, the language model generated in the previous step is adapted to the new domain using the “Sherlock Holmes” texts. The adapted model is produced by combining the existing business news model (weight 0.75) and a new language model implicitly generated from the text data (weight 0.25).

```
LAdapt -C ../prune.cfg -S lmdata.scp -T 1 -w holmes.wlist \
-i 0.75 models/holmes_BN1_med.lm models/holmes_med.lm
```

6. **Evaluate the “goodness” of the new models.** The produced models are evaluated by computing the perplexity of previously unseen text from the target domain. First, the perplexity of the “Red-headed league” text is evaluated using the business news model.

```
LPlex -T 1 -t models/holmes_BN1_med.lm test/red-headed_league.txt
```

```
Loading language model from models/holmes_BN1_med.lm
Using language model(s):
  3-gram models/holmes_BN1_med.lm, weight 1.00
Found 20091 unique words in 1 model(s)
LPlex test #0: 3-gram
Processing text stream: test/red-headed_league.txt
perplexity 296.3688, var 12.4454, utterances 556, words predicted 9329
num tokens 10408, 00V 189, 00V rate 1.92% (excl. </s>)
```

Next, the test set perplexity is evaluated using the adapted model.

```
LPlex -T 1 -t models/holmes_med.lm test/red-headed_league.txt
```

```
Loading language model from models/holmes_med.lm
Using language model(s):
  3-gram models/holmes_med.lm, weight 1.00
Found 20091 unique words in 1 model(s)
LPlex test #0: 3-gram
Processing text stream: test/red-headed_league.txt
perplexity 173.1044, var 9.9893, utterances 556, words predicted 9329
num tokens 10408, 00V 189, 00V rate 1.92% (excl. </s>)
```

The adapted model gives a much improved perplexity figure of 173 over the original 296 given by the business news model.

7. **Create a system configuration file.** The recogniser configuration file for use with the System evaluation tool is given below.

```

## UK Sherlock Holmes recogniser
## British English models, "Sherlock Holmes" 20K vocabulary,
## medium size language model derived by adapting the business
## news model using the Sherlock Holmes texts

# Default is no tracing - use suggested values to track problems

HAPI:      TRACELEVEL = 0   # 07777
HAudio:    TRACE = 0       #      3
HParm:     TRACE = 0       #     022
HNet:      TRACE = 0       #      1
HLM:       TRACE = 0       #      1

# models and other resources
HAPI:      MMF           = resources/EPCX2_UK01.mmf
HAPI:      HMMLIST      = resources/EPCX2_UK01.list
HAPI:      DICTFILE     = examples/holmes/models/holmes_uk.dct
HAPI:      NETFILE      = examples/holmes/models/holmes_med.lm

#include "hapilvx.cfg"
#include "coding16epc.cfg"
#include "adapt_uk.cfg"
#include "audio.cfg"

```

The above HAPI configuration file for the “Sherlock Holmes” recogniser is available in `config/holmes_uk.config`.

Chapter 3

Advanced adaptation techniques

The TRANSCRIBER distribution features adaptation in various forms – the Enrollment tool demonstrates offline supervised adaptation, whilst online supervised adaptation (i.e. the transcribed speech text can be corrected before adaptation) and online unsupervised adaptation are available in the System evaluation tool. All these forms of adaptation use class-based transforms.

To generate a more specific and accurate speaker-dependent model, which has undergone the full transformation, the command line tool HAPIADAPT should be used. Before running HAPIADAPT, some speech data must be collected. The easiest way to do this is to use the TRANSCRIBER enrollment procedure. Not only will this produce a class-based TMF, but it will also store all the waveforms, and a speech data script file in the user’s session directory.

As an example, it will be assumed that some data has been recorded by the user “djk” during a “Sherlock Holmes” enrollment session, and that it resides in the `sessions/djk01` folder (see section 2.1). This folder contains a file called `djk01.scp`. This is the speech data script file, which contains a list of all the speech files recorded during the enrollment. The folder also contains another folder called `waveforms`, which holds all the recorded waveforms. In all subsequent examples, the command line tools should be executed from the TRANSCRIBER directory.

The adaptation command line tool (HAPIADAPT) is available in the `bin.$CPU` directory. Typically, HAPIADAPT requires a configuration file. Such files are supplied in the `config` directory for each of the supported languages. The user may alter the parameter settings used during adaptation by editing the appropriate configuration file. Furthermore, some settings can be overridden from the command-line. A full reference section for HAPIADAPT can be found in section 6.1.

3.1 Supervised adaptation

For supervised adaptation, the speech data transcriptions are available and can be found in the `prompts` directory for the Sherlock Holmes (The Red Headed League) story. The prompt file `red-headed_upcase.pmt` contains the transcription in the required format for HAPIADAPT. Also contained in this directory are dictionaries which contain all the words in the Sherlock Holmes story. It is up to the user to pick

the US, UK or UX dictionary. For the user “dj k”, an example usage for HAPIADAPT is:-

```
HAPIAdapt -C config/hapiadapt_uk.cfg -p prompts/red-headed_upcase.pmt -B
-T 1 -u dj k -d prompts/red-headed_uk.dct -S sessions/dj k01/dj k01.scp
-M sessions/dj k01
```

The configuration file `config/hapiadapt_uk.cfg` contains the standard configurations for a UK user. (A US and UX equivalent can also be found in the `config` directory.) Note in the above command line how the default dictionary in the configuration file is overridden by specifying a primary dictionary. After running this command (process time is dependent on the amount of data), two files will be produced in the `sessions/dj k01` directory. The first file will be called `dj k.tmf` and the second `EPCX1_UK01_dj k.mmf`. The first file contains a class based transform set, which can be applied to the speaker-independent model set `EPCX1_UK01.mmf`. The second file is a “speaker-dependent” MMF solely for user `dj k`. Both files are written in binary format (which is far more compact than the ASCII equivalent).

The next two command-lines give examples of how the adaptation procedure may be run in the case where no specific dictionary is available.

Example 1:-

```
HAPIAdapt -C config/hapiadapt_uk.cfg -p prompts/red-headed_upcase.pmt -T 1 -B
-u dj k -S sessions/dj k01/dj k01.scp -M sessions/dj k01
```

Example 2:-

```
HAPIAdapt -b -C config/hapiadapt_uk.cfg -p prompts/red-headed_upcase.pmt -T 1
-B -u dj k -S sessions/dj k01/dj k01.scp -M sessions/dj k01
```

In example 1, if one or more words in the transcription of an utterance is not found in the system’s vocabulary (as specified in the configuration file), that utterance is skipped for the purposes of adaptation. This method could potentially throw away a lot of important speech data. In example 2, the option `-b` is specified. This instructs HAPIADAPT to use the background/garbage word model for words in the transcription which are not present in the system’s vocabulary. Thus only a small amount of speech data is thrown away (i.e. all the data that aligns to the background word model, if and when it is used). For the whole of the Sherlock Holmes story, when using the default dictionary and a background word model, the background word model is only used for 113 of the 9209 words.

3.2 Unsupervised adaptation

In this case a transcription is not available for the adaptation process. Unsupervised adaptation is triggered when a prompt file is unspecified. Note that this unsupervised mode performs recognition in order to acquire a transcription for each utterance, and hence the whole adaptation process takes much longer. An example command-line is shown below...

```
HAPIAdapt -C config/hapiadapt_uk.cfg -T 1 -B -u dj k -S sessions/dj k01/dj k01.scp
-M sessions/dj k01
```

3.3 Testing the new TMF and MMF

On completion HAPIADAPT produces both a class-based transform, saved in a TMF, and a fully transformed model set saved in an MMF. Both these new models can be tested using TRANSCRIBER.

In order to evaluate the TMF, start up the System evaluation tool and select the appropriate configuration which includes the speaker-independent MMF and list files. Once the recogniser has finished loading the acoustic model set, dictionary and language model, it is ready for testing. Select the **load transform** option from the **adaptation** ↓ pull down menu. A file dialog will be displayed. Navigate to the correct transform model file and select it (i.e. `sessions/djk01/djk.tmf`). The adaptation transform will be loaded and applied to the acoustic model set. The recogniser is now ready for use.

For a fully transformed model set, a new configuration file must be constructed. To start, move to the `config` directory and copy an appropriate configuration file (which will be used as a template) to a new file. For instance if user “djk” has generated a new model set called `EPCX1_UK01_djk.mmf` then a copy should be made of the configuration file `60nv_uk_comp.config` to something like `60nv_djk_comp.config`. Load the new configuration file into a text editor and firstly change the comment lines at the top of the file to reflect the fact that this is a configuration setup solely for user “djk”. The only other change necessary is the following line to incorporate the new speaker-dependent MMF:-

```
HAPI      : MMF          = resources/EPCX1_UK01.mmf
           to
HAPI      : MMF          = sessions/djk01/EPCX1_UK01_djk.mmf
```

Save these changes in the new configuration file, and restart TRANSCRIBER. Select the new configuration file on startup, and once the model set, dictionary and language model have been loaded, it is ready for use with this new acoustic model.

3.4 Adaptation types

There are three different transformation types¹ available to the developer, namely global, class-based and full transformations. The global transformation is generally used in the first pass of a multi-pass adaptation training scheme to provide an improved alignment for the subsequent adaptation pass. Any final transformation that the user has access to should use the class-based transform or the fully transformed MMF, since these transformations are superior to the global transformation in terms of speed and accuracy. Table 3.1 makes a comparison between the class-based and full transformation, highlighting their strengths and weaknesses respectively. It is up to the developer to decide which kind of transformation is appropriate for their particular application.

Another important feature is that the class-based and full transformation methods can be combined (by just setting both for the transform apply type). This transformation normally results in superior performance when compared to solely using either the class-based or full transformations, no matter how much adaptation material is

¹For more information on speaker adaptation the reader is referred to the HAPI Book.

Feature	Class	Full
Adaptation performance after a few minutes of data	Fast adaptation is possible, and improvement in terms of speed and accuracy is noticeable.	Adaptation likely to be ineffective.
Adaptation performance after twenty minutes of data	Improvement in terms of speed and accuracy is very noticeable.	Improvement in terms of speed and accuracy is very noticeable and comparable with class-based.
Adaptation performance after an hour or more of data	Improvement over transforms estimated from twenty minutes tends not to be especially evident. Adaptation process is likely to have saturated.	Adapted model performance continues to improve, and is likely to be better than the classed-based method.
Saving models	There are options to save a TMF (which is considerably smaller than an MMF) or an MMF. The TMF can also be stored with statistics, so that adaptation can resume from a saved point.	It is only possible to save the MMF. Adaptation can be continued from a newly saved MMF, but care must be taken when setting the <code>FULLADPTSCALE</code> configuration parameter so as not to discount previously seen data that was used in the loaded MMF.
Flexibility	For a new speaker it is easy to apply/revert a transform, and this process is fast, and can be done many times.	For a new speaker a whole new model set must be loaded, which is a little more time consuming than class-based.

Table 3.1: A comparison of the class-based and full transformations.

used. However since a full transformation is made, the developer must save a full model set.

3.5 Choosing an appropriate model set

TRANSCRIBER currently provides the acoustic model sets described in appendix A. An example of using the UK compact model set can be found in the configuration file `config/hapiadapt.uk.cfg`. All supplied models are speaker independent, and can be used for speaker and/or environmental adaptation purposes. The configuration files provided with HAPIADAPT specify the compact models as the default model set. However, this can either be overridden using the appropriate command-line options, or by editing the configuration file to choose the model set required.

In general, before any adaptation, the larger the model set, the more accurate the performance, but this comes with a slight increase in processing time required. For longer term adaptation – i.e. more than 3-4 hours of adaptation material, it may prove more beneficial to switch to a larger model set. Again, it is up to the application developer to determine which model is most suited to the application's requirements and specifications.

3.6 Guide for incremental adaptation

Incremental adaptation is where the model set or transform set is continually refined on an incremental basis – perhaps as often as every utterance or as little as every user session, in which case it is usually referred to as block adaptation.

3.6.1 Incremental class-based adaptation

After enrollment the system can be configured to continue to adapt the model set in an incremental fashion while it is being used for recognition purposes. This is currently demonstrated by TRANSCRIBER where a class-based transform that has been saved with statistics is continually tuned to the user during the recognition processing. This adaptation mode is online supervised adaptation as described in the HAPI book, where rather than starting adaptation from scratch, an existing TMF (with stored statistics) is loaded and adaptation continued from this point. At the end of the session a new TMF can be saved.

3.6.2 Incremental full adaptation

This feature is not demonstrated currently by TRANSCRIBER. For incremental full adaptation, a model set is loaded, adapted and finally a new model set is saved. The developer is strongly advised to use block adaptation only (i.e. update the model set only at the end of the session, before saving a new MMF). As more and more data is collected, the developer is also advised to start gradually increasing the `FULLADPTSCALE` configuration parameter.

3.7 Initialising models for adaptation

All models supplied with TRANSCRIBER come pre-initialised for adaptation. This section is solely for the benefit of HTK users, and should otherwise be ignored.

If a developer has already built his/her own acoustic model set, and wishes to integrate this with TRANSCRIBER and/or any applications that make use of the adaptation facility, then the acoustic model set must be initialised for adaptation. The HTK tool, HHED must be used for this. Note that this is the HHED tool supplied with TRANSCRIBER and is different to the standard HTK version. The following HHED script, called `hhedfile`, will assign each model set component to one of 32 different classes, while also renaming the MMF identifier. The `STATSFILE` is the occupation statistics file that is output after a training pass with HEREST.

```
RN "InHouseMMF_UK"  
LS "statsfile"  
RC 32 "InHouseMMF_UK_rtree" sil.state[2-4].mix
```

A maximum of 256 classes is possible. The following HHED command makes use of this script file to construct a new MMF which can then be used for adaptation.

```
HHed -T 1 -B -H old.mmf -w new.mmf hhedfile list
```

Further information is also available in the reference section for HHED, together with other HTK tools supplied with TRANSCRIBER.

Part II

Transcriber tools

Chapter 4

System evaluation tool

The system evaluation tool is a JHAPI application which allows the user to interactively evaluate the performance of a large vocabulary recognition system. The tool is started by changing to the TRANSCRIBER directory and executing the batch script `lvrdemo`.

4.1 A tour of the main window

The main window consists of a menu bar and four panels - *Controls*, *Output*, *Current* and *Status*.

4.1.1 The menu bar

The tool's menu bar consists of four menus. The **File** menu allows the user to configure and load the recogniser. The following options are available:

- **Configure recogniser**
Choosing this item brings up the recogniser configuration dialog (see figure 4.1). The user is prompted to specify the recogniser configuration and session parameters.
- **Load recogniser**
Choosing this item loads the recogniser's components and prepares the system for recognition.
- **About**
Display TRANSCRIBER release information.
- **Quit**
Exit the system evaluation tool. Note that this option is available at any time except during the loading of the recogniser's components.

The **Session** menu becomes active once the recogniser components have been loaded. It allows the user to configure the data source, toggle the recognition output mode and calibrate the speech/silence detector.

- **New session**

Initialises a new session. Brings up the session configuration dialog and prompts the user for session directory, session ID and gives the options to save label files and lattices. After the new session is configured, the screen is cleared and any speaker adaptation data accumulated so far is discarded. For optimal performance this should be carried out for each new speaker as it resets the information used to perform speaker normalisation to a default value. If this is not performed the initial few seconds of speech from a new speaker may not be recognised as accurately as the remainder.

- **Reset Screen**

Rebuild the recognition output in the *Output* panel from the most likely recognition hypothesis for the 16 most recent utterances recognised. This operation will discard all text edits performed manually or via the N-best facility. The currently selected output mode will be used.

- **Word-output mode**

This menu item toggles the output mode used to display recognised text. By default, recognition output is displayed using the output symbols specified for each word in the vocabulary. Note that supervised adaptation requires the user to perform editing of the recognised text in word output mode. Switching between modes will cause any changes made to the text so far (manually or using the N-best facility) to be lost.

- **Source settings**

Select the source providing the speech data for recognition. A dialog appears on the screen allowing the user to select one of the following sources: **Mic-in**, **Line-in**, **File** and **Script**. In addition the speech detector can be enabled or disabled.

- **Calibrate recogniser**

Calibrate the speech/silence detector parameters in the current acoustic environment. Calibration is normally required when the speech/silence detector is used in conjunction with direct audio sources such as **Mic-in** and **Line-in**. Although this procedure is only used to set the parameters of the speech detector (assuming it is being used), the values displayed are useful for checking that the audio equipment is set-up correctly. Every time calibration is performed, the silence threshold value (**SILENERGY**) used by the detector will be stored in the file `config/audio.cfg`. This setting will be automatically used until another calibration is performed. In shared environments where a single copy of **TRANSCRIBER** is installed but used on multiple workstations, it is advised that calibration is performed every time the system evaluation tool is started.

The **Utterance** ⇓ menu provides options for utterance playback and reverting to the most likely utterance hypothesis.

- **Reset Utterance**

Reset utterance transcription to most likely hypothesis from recogniser.

- **Replay**

Replay the selected words or the whole utterance.

The **Adaptation** ↓ menu allows the user to load and apply speaker adaptation transforms, display transform information and start up the Enrollment tool.

- **Load transform**

This option allows the user to load and apply an adaptation transform to the current acoustic model set. A file dialog is displayed which enables the user to locate the desired transform model file (TMF). Once loaded, the transform can be further updated using the recognition output in both supervised and unsupervised mode.

- **Transform info**

Display information about the loaded adaptation transform. Click **OK** to dismiss.

- **Unload transform**

Transform the acoustic model back to its original state. A dialog is displayed asking the user to confirm the operation.

- **Adapt model (unsupervised)**

A new adaptation transform is calculated from the current recognition output and applied to the acoustic model. The most likely recognition hypothesis for each utterance is used as a reference transcription in the adaptation. Corrections made to the recognised text are not taken into account.

- **Adapt model (supervised)**

A new adaptation transform is calculated from the currently displayed recognition output and applied to the acoustic model set. This operation is normally used after the user has corrected any errors in the recognised text either manually or using the N-best facility. All corrections must be performed in **Word-output** mode.

- **Save transform**

Allows the user to save the current adaptation transform.

- **Auto adapt**

Enables automatic unsupervised adaptation after a specified number of recognised utterances.

- **New enrollment**

This starts up the Enrollment tool and creates a new enrollment session.

- **Resume enrollment**

This starts up the Enrollment tool and allows the user to continue an existing enrollment session.

4.1.2 The Controls

The *Controls* panel contains four buttons used to start/stop recognition, obtain N-best alternatives, replay recorded utterances and perform model adaptation.

Start, **Stop**, **Abort**

This button is used to control the operation of the recogniser, indicating when it should start, stop and abort processing. Its label reflects its current function based on the state of the recogniser.

N-best

Clicking this button displays a list of alternative recognition hypotheses for the selected word sequence or the whole utterance.

Replay

Replay the currently selected word sequence or the whole utterance.

Adapt

Perform model adaptation using the most recently recognised speech utterances. If the recognised text is displayed in **Word-output** mode, supervised adaptation will be performed. Otherwise, unsupervised adaptation will be used.

4.1.3 The Output Panel

This text area is used to display the output of the recogniser. The text can be edited using the keyboard and mouse as well as by using the facilities for N-best alternatives. Note that the text window is not a fully fledged word processor and provides only simple editing facilities (such as scroll bars and selections but no word wrap or saving facilities). The text can be copied to a normal word processor using the cut and paste facilities provided by the windowing system (WINDOWS/X11).

4.1.4 The Current Panel

Towards the left hand side of this area status messages are displayed, while the meter on the right can act as both a volume level meter during recognition and as a progress indicator during loading.

4.2 Configuring the recogniser

The LVX decoder can be configured to perform a variety of tasks by using different models, vocabularies and setups. Each task is defined by the acoustic model set, pronunciation dictionary and language model used by the recogniser. Choosing a different set of acoustic models could optimise performance for a particular dialect, while using a different language model would optimise the recogniser for a different target domain.

4.2.1 System configuration files

A recogniser configuration for a particular task is described in a HAPI configuration file. Configuration files for the example systems supplied with TRANSCRIBER reside in the `config` directory. As a general rule, the names of config files created by the user should end in `.config`. The following is the HAPI configuration file for the 60K UK system.

```
## UK 60NV dictation
## British English models, 60K VP vocabulary, medium business news
## dictation language model

# Default is no tracing - can use suggested values to track problems
```



```

HAPI:      TRACELEVEL = 0   # 07777
HAudio:    TRACE = 0       #      3
HParm:     TRACE = 0       #    022
HNet:      TRACE = 0       #      1
HLM:       TRACE = 0       #      1

# models and other resources
HAPI:      MMF            = resources/EPCX2_UK01.mmf
HAPI:      HMMLIST       = resources/EPCX2_UK01.list
HAPI:      DICTFILE      = models/60nv_uk.dct
HAPI:      NETFILE       = models/60nv_med.lm

#include "hapilvx.cfg"
#include "coding16epc.cfg"
#include "adapt_uk.cfg"
#include "audio.cfg"

```

The first line in the file gives the name of the system. This line is only read and interpreted by the system evaluation tool. The system name is displayed in the selection list of the recogniser configuration dialog after selecting **Configure Recogniser**. Any subsequent lines starting with **##** are interpreted as the description of the system and displayed in the scrollable text area in the configuration dialog. The first block of configuration parameters sets the tracing for the various software components in HAPI. For more information on tracing options refer to the HAPI Book. The second block of configuration parameters defines the components of the recogniser for the task - the acoustic model file name, the HMM model list, the dictionary file and the language model. Finally, four extra config files are included. The `hapilvx.cfg` file contains configuration parameters for the LVX decoder. The `coding16epc.cfg` file sets the EPC coding parameters for 16KHz waveform data. The file `adapt_uk.cfg` defines the parameters used during speaker adaptation. Finally, the `audio.cfg` file contains the value of the `SILENERGY` parameter from the last calibration of the speech/silence detector.

4.2.2 Choosing a system configuration


In the system evaluation tool, a particular recogniser configuration is chosen via the **Configure Recogniser** option from the **File**  menu. The system configuration dialog is displayed listing the available system configurations together with their description (see figure 4.1). When the dialog is displayed, the tool inspects the default config directory (`config`) and forms a list of files whose names end in `.config`. The system name and description are read from each config file by inspecting the first few lines which start with **##**. A list of available systems is formed and displayed in the choice list area. To load configurations from elsewhere, type the name of the directory in the **Config directory** field and click **Rescan**. Any system configurations available in that directory will be displayed in the choice list. To select a system configuration click on the system name in the choice list. Information about the system will be displayed in the text area below.



Fig. 4.1 The system configuration dialog.

At this stage, the system evaluation tool can be configured to save various information whilst performing recognition such as recognition results (in a master label file), recognition lattices which contain multiple sentence hypotheses and the waveform files themselves. The output of the recogniser is saved in a session directory specified by the user. To save output from the recogniser click on the **Save session** check-box in the bottom half of the configuration dialog (default setting). This enables the following further settings:

- **Save lattices**
Tick (✓) this box if you wish recognition lattices to be saved.
- **Save MLF**
Tick (✓) this box if you wish recognition results to be saved.
- **Session ID**
This is the session ID for the recognition session. The default ID of the first session is `rsn001`.
- **Output directory**
This is the output directory where the session will be saved to. The default value is `sessions`.

All recognition output will be saved to the session directory. In the following description it will be assumed that the session output directory is `sessions` and the session ID is `rsn001`. The system evaluation tool will automatically create a new directory `sessions/rsn001`. Waveform files will be labelled `rsn001uNNN.wav` and saved to the `waveforms` subdirectory where `NNN` is the numeric utterance ID (starts from 000). Lattices will be saved in the `lattices` subdirectory and named `rsn001uNNN.lat`. Recognition results will be saved to the Master Label File (MLF) `rsn001.mlf`. Note

that if the recognition is performed from a script file rather than from direct audio input, waveform data will not be saved and the name of each utterance will be derived from the corresponding name in the script file.

Note that even when session saving is disabled, all utterances are temporarily stored on disk in order to allow for selective playback. Permanent recording of utterances is useful for both testing of future recognisers as well as adapting a recogniser to the specific user. Audio recording consumes a great deal of disk space (over 100Mb for each hour of speech) so unless there is a specific need for the waveform files there is no need to keep a permanent copy.

4.3 Loading the recogniser

The recogniser components are loaded in two stages. The acoustic models (HMMs) and the dictionary are loaded immediately after the recogniser configuration has been chosen and the configuration dialog dismissed. Once loading has completed, the user may choose to start up the *Enrollment* tool or load the remaining component(s) and start the recogniser.

Choosing **Load Recogniser** from the **File** ↓ menu will load the language model. This may take several minutes with progress indicated in the status area at the bottom of the window. The recogniser is ready for use when the message *Loading Done* is displayed and the red progress dot stops moving and changes colour to blue.

4.4 Source settings and calibration

The recogniser can be configured to recognise utterances from a number of sources. In addition, when recognising data from direct audio input, utterance boundaries can be marked automatically by a speech/silence detector or manually by the user. The source settings options are set from the **Source Settings** dialog from the **Session** ↓ menu. The available sources are:

- **Mic-in**

In this mode data will be acquired from the machine's built-in audio device via the microphone input. Automatic speech/silence detection or manual operation can be used.

- **Line-in**

In this mode data will be acquired from the machine's built-in audio device via the line input. Automatic speech/silence detection or manual operation can be used.

- **File**

Recognition will be performed from a waveform or parameterised data file. Upon selection, a file dialog is displayed prompting the user to locate the desired data file.

- **Script**

Recognition will be performed from a waveform or coded data files whose names are listed in the specified script file.

The default source setting is direct audio input via the microphone port with automatic speech/silence detection enabled. Note that under Windows9x/NT, the **Mic-in** and **Line-in** settings are identical. The actual audio input can be configured from the *Volume Control* accessory available from the start menu.

The recogniser is automatically configured to use a speech/silence detector to delimit the period of speech that comprises each utterance. Although the recogniser itself is speaker independent and able to cope with a variety of background and speaking levels, the silence detector needs an estimate of the ambient background noise level in order to function correctly. It is also useful to be able to measure the speech/silence signal levels when initially setting up input levels and amplifier gains.



Fig. 4.2 Speech/silence detector calibration dialog.

The **Calibrate Recogniser** menu option displays a dialog box with options to accomplish both of these functions. To calibrate the speech detector, click on the **Calibrate** button and speak for 5 seconds. The following measured values will be displayed:

- **Background level** - This is the background silence level. When the silence detector is being used, the silence level should be between 15 and 30dB. If the speech detector is disabled, n/a is displayed.
- **Mean speech level** - The mean speech level should be 30 to 40dB higher than the measured silence level to maintain recogniser accuracy (around 55 to 70dB). If the speech detector is disabled, n/a is displayed.

- **Dynamic range** - This measurement reflects the dynamic range of speech signal. The value should be less than 75% to ensure that the signal is not clipped. If the speech detector is disabled, n/a is displayed.

The calibration procedure must be carried out when the system evaluation tool is started for the first time. The levels should be carefully checked to ensure that the audio input is set up correctly. If the background noise level changes during recognition, the procedure should be repeated to ensure that the silence detector continues to work accurately.

If the signal to noise ratio - mean speech level minus the silence level (both in dB) - is close to 0 dB check that the microphone is connected correctly and the amplifier is switched on. If the levels are too high (peak-to-peak more than 75%) or too low (mean speech less than 40dB) check and adjust the gain of both the pre-amplifier (using its controls) and of the computer's audio input by running the *Volume Control* accessory in WINDOWS.

4.5 Performing recognition

Once the input levels have been checked (and the speech detector calibrated) the recogniser is ready for use. The *Controls* panel contains three mouse operated buttons controlling the recogniser. The recogniser can be operated in speech/silence detection mode or push-to-start push-to-stop mode. Recognition is started by clicking the **Start** button. Subsequent control of the recognition process depends on the operation mode.

- **Speech/silence detection mode.** When **Start** is clicked, (and *Recognising* appears in the status line at the bottom of the window) the silence detector is activated and waiting for the user to begin speaking. As soon as the onset of speech has been detected the VU meter in the lower right corner of the screen will indicate the input speech level and the *Current* panel will show the most likely partial hypothesis for the unknown speech.

Having detected the onset of speech the decoder will start processing the audio input and the button label changes to **Abort** providing the user with the facility to terminate recognition before the end of the utterance has been detected.

If the **Abort** button is not pressed the recogniser will continue processing the speech until the silence detector determines that the utterance is finished (indicated by 1 second of silence). At the same time the VU meter will drop to its minimum level whilst the recogniser finishes processing the utterance.

- **Push-to-start/push-to-stop mode.** In this mode the user explicitly marks the boundaries of speech to be processed by the recogniser.

After clicking the **Start** button, the recogniser immediately begins processing the audio input and the button label changes to **Stop**. When clicked again, audio input stops. The recogniser continues to process the remaining part of the utterance (although this can be stopped by pressing the button - now labelled **Abort** - for a third time).

When the whole utterance has been processed the most likely complete utterance hypothesis will be written to the *Output* panel and the button label will revert back to **Start**. Whilst audio input is active the VU meter in the lower right corner of the screen will indicate the current speech level. At the same time, the *Current* panel will show the most likely hypothesis for the input speech data.

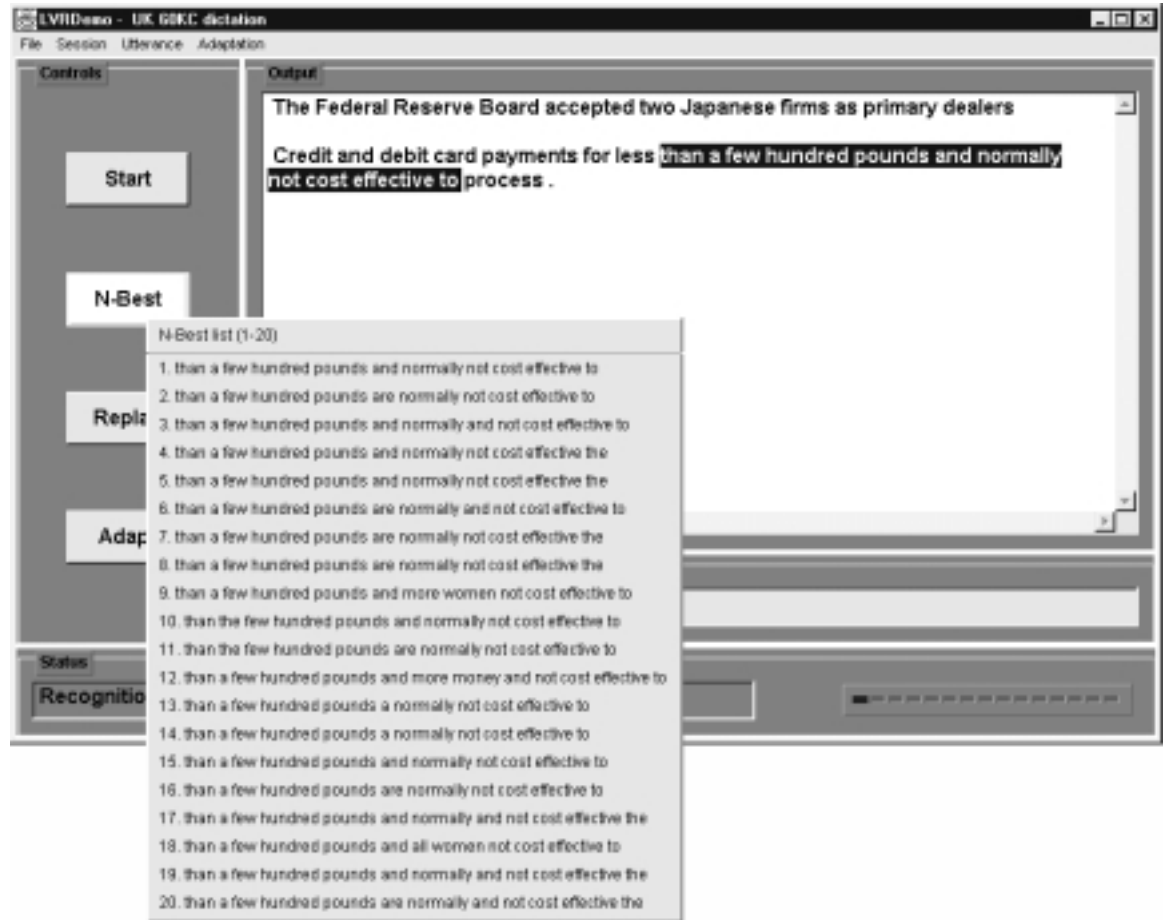


Fig. 4.3 N-best results.

4.6 N-best results and playback

Once recognition output is available, clicking the **N-best** button will display a list of alternatives for the current selection¹. An example of using this facility to display a list of alternatives for a particular word sequence is shown in figure 4.3.

¹A “selection” is either a highlighted word sequence or the complete utterance within which the text cursor is currently placed.

The N-best facility allows easy correction of the utterance transcription when the recogniser has included the correct word sequence in the lattice of alternatives that have been considered to closely match the speech. Note that if the user edits the text to form a sequence of words which does not represent a possible path through this lattice of alternatives, the N-best list will not be available and the message *Unable to parse text* will be displayed instead. If substantial manual editing takes place then the text parser may not be able to find the word sequence representing each utterance and will never be able to find alternatives. If this happens, the user can choose the **Reset screen** option from the **Session** ↓ menu to discard all edits made so far and revert to the most likely recognition hypotheses. This operation will recover the N-best functionality. Note that the N-best facility is limited to the 16 most recently recognised utterances.

Clicking the **Replay** button will play the current selection (either the highlighted words or the complete utterance). This allows the user to verify that the speech detector operated correctly, that the hypothesis agrees with the spoken words and that the word boundary times have been recognised correctly.

4.7 Speaker/environment adaptation

The system evaluation tool incorporates the facility to load acoustic model transformations which will adapt the model parameters to a particular speaker and/or environment characteristics. Adaptation transforms reside in Transform Model Files (TMFs) generated by the Enrollment tool or the HAPIAdapt tool in TRANSCRIBER. Note that TMFs are model-specific, hence attempting to load a transform which does not match the acoustic model currently used by the recogniser will result in error.

To load an adaptation transform, select the **Load transform** option from the **Adaptation** ↓ menu. A file dialog will be displayed prompting the user to locate the TMF. When done click **Open/OK**. An information dialog will be displayed whilst loading the transform. Once loaded, the transform will be automatically applied to the acoustic model set. Any subsequent recognitions, will use the adapted model set.

If used correctly, speaker/environment adaptation should result in improved recognition speed and accuracy.

To obtain information about the currently loaded transform select **Transform info** from the **Adaptation** ↓ menu. An information dialog will be displayed giving user name, acoustic channel details, etc. Click **OK** to dismiss.

To unload the current transform, select **Unload transform** from the **Adaptation** ↓ menu. A warning dialog will appear asking the user to confirm the operation. If confirmed, the current adaptation transform will be unloaded and the acoustic model will revert back to its original state.

4.7.1 Supervised and unsupervised adaptation

Adaptation transforms are produced from speech data collected from the target user in the desired acoustic environment. Transforms are applied and calculated relative to a speaker independent model set e.g. it is not possible to adapt an already transformed model set. At the same time, an adaptation transform can be updated as more speaker-specific data becomes available.

In particular, recognised utterances can be used to further refine a currently loaded adaptation transform or create a new transform from scratch. The System Evaluation tool provides two methods for refining or creating adaptation transforms. In *unsupervised* mode model transformation is performed using the recognition output as a reference transcription. In *supervised* mode the user is expected to manually correct any errors in the recognition output. The corrected text is then used as a reference transcription during model adaptation. In order to perform *supervised* adaptation, the user must select **Word-output** mode from the **Session ↓** menu. In this mode the output text is displayed using the actual words in the vocabulary as opposed to the output symbols specified for each word. Subsequently, manual corrections of the recognition output should use the same word symbols as specified in the dictionary. Furthermore, the sentence start (<s>) and sentence end (</s>) markers should be used to mark the boundaries of each utterance.

Supervised and unsupervised adaptation using the recognised speech data are available from the **Adaptation ↓** menu. When selected, the current recognition output is scanned backwards from the most recently recognised utterance, a new adaptation transformation is computed and applied to the acoustic model set. In addition, the **Adapt** button in the **Controls** panel can be used to trigger adaptation. This will perform supervised adaptation if recognition is currently displayed in **Word-output** mode, unsupervised adaptation will be used otherwise. The **Auto adapt** option from the **Adaptation ↓** menu can be used to enable automatic unsupervised adaptation after a specified number of newly recognised utterances.

The currently computed adaptation transform can be saved for future use using the **Save transform** option from the **Adaptation ↓** menu. A file dialog will be displayed prompting the user to specify the target directory. When done click **Save/OK**.

The TRANSCRIBER package also incorporates an Enrollment tool which can be used to collect speech waveform data and generate adaptation transforms. To invoke the Enrollment tool, select the **New enrollment** option from the **Adaptation ↓** menu (see figure 4.4). The Enrollment tool's main window will appear on the screen. Details on how to use the Enrollment tool are given in chapter 5.

4.7.2 Notes

- The status area contains a message window which informs the user of the current action being carried out. The progress bar displays a single red moving dot during recogniser loading. This changes into a volume meter during recognition, which can be used to determine the state of the silence detector.
- Do not start speaking until the message *Recognising...* appears in the status area at the bottom left corner of the window. Note that the **Abort** button is inactive until the recogniser has started to generate output in the *Current* panel. This means that when the speech/silence detector is enabled the user will have to start speaking in order to abort the recognition.

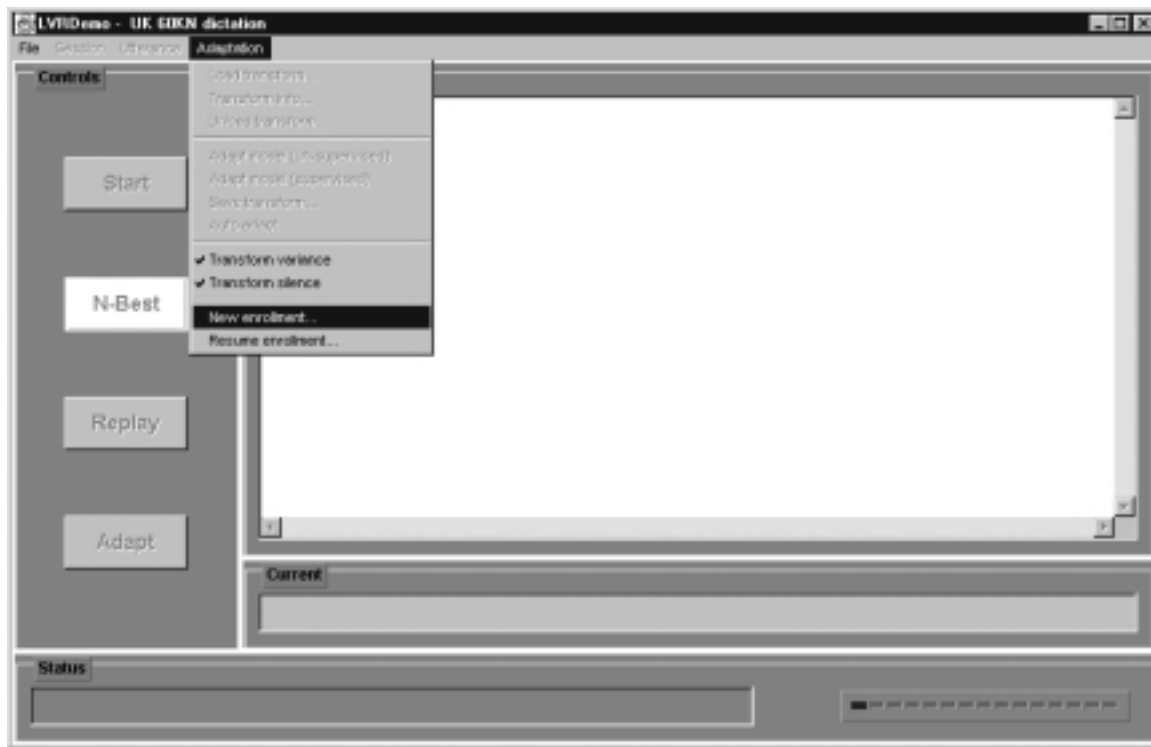


Fig. 4.4 Starting up the Enrollment tool.

Chapter 5

Enrollment tool

The Enrollment tool enables the user to record speech data by reading out a sequence of utterance prompts. The recorded data is then used to compute a speaker adaptation transform¹ for a particular acoustic model set. The recorded waveform data and other information such as speaker details, data list and adaptation configuration files are stored in a session-specific directory hierarchy. The tool allows a recording session to be terminated at any point and resumed at another time. The Enrollment tool is accessible from the **Adaptation ↓** menu of the System Evaluation tool once the user has chosen a particular system configuration. The speech data collected using the Enrollment tool can be used also in creating speaker dependent model sets via the `HAPIAdapt` command line tool in `TRANSCRIBER`.

5.1 A tour of the main window

The main window consists of a menu bar and three panels - waveform display, prompt area and control buttons/status area (see figure 5.1).

5.1.1 The menu bar

The menu bar consists of two menus. The **File ↓** menu allows the user to create a new enrollment session, load an existing session or shutdown the tool.

- **New session**

A new enrollment session is created. The session configuration dialog is displayed and the user is prompted to fill-in general information about the speaker and the acoustic environment. Note that the speaker name must not be left empty since it will be used to derive the speaker ID encoded in the generated adaptation transform. The default session ID is set to `sn000`. However, the user is advised to enter a meaningful identifier. The output directory specifies the location where the session directory will be created. A new enrollment session is automatically created when the tool is invoked with the **New enrollment** option from the System evaluation tool.

¹The speaker adaptation facilities provided by the Enrollment tool are implemented entirely via the `HAPI` (`JHAPI`) library.

- **Load session**

Resume an existing enrollment session. A file dialog is displayed asking the user to specify the session information file. Note that the session files generated by the Enrollment tool end with the extension `.inf`.

- **Save session**

Save the current enrollment session. The session information will be saved to `<dir>/<id>/<id>.inf` where `<dir>` is the output directory set in the session configuration dialog and `<id>` is the session ID.

- **Parameters**

Alter the current session information. The session configuration dialog is displayed allowing the user to change the current session information. Note that if any waveform data has been recorded already, the data in the following fields cannot be altered: output directory, session ID and waveform extension.

- **Close**

Close the Enrollment tool and exit. The user will be prompted to save any changes made to the current session.

The **Settings** ↓ menu has the following options

- **Auto increment**

This option toggles the auto increment mode on and off. In auto-increment mode, the enrollment procedure requires minimum control from the user. In this mode the user is asked to record each prompt in turn. If the utterance is recorded correctly, the current prompt is automatically advanced. Once all prompts have been recorded, an adaptation transform is calculated.

- **Compact TMF**

If “Compact TMF” is selected, the saved adaptation transform will be smaller in size, however, such transforms cannot be further updated as more adaptation data becomes available.

- **Alignment beam**

This option allows the user to set the general alignment beam used to align the spoken utterance against the text prompt. The alignment procedure is carried out in order to check if the spoken utterance matches the prompt and reject speech unsuitable for adaptation. In general, small beam values are likely to cause the alignment procedure to fail even for minor mispronunciations, non-speech events, etc. In such cases the user will be asked to repeat the recording.

- **Confidence Threshold**

This option allows the user to set the confidence threshold used in deciding if a prompt has been successfully recorded. Once the recorded utterance has been successfully aligned against the prompt, a measure of confidence is calculated for each overlapping word triplet in the utterance. If a confidence score falls below the specified threshold, the utterance is marked as unsuitable for adaptation purposes and the user is asked to re-record it. The confidence threshold value should be in the range (0.0 - 1.0). Setting the threshold to 0.0 disables confidence score based rejection.

- **Source settings**

Select the source of audio data. A dialog appears on the screen allowing the user to select one of the following sources **Mic-in** and **Line-in**. In addition the speech detector can be enabled or disabled. Note that under **WINDOWS** the source setting has no effect and the input source should be selected using the *Volume Control* accessory.

- **Calibrate source**

Calibrate the speech/silence detector parameters in the current acoustic environment. Calibration is normally required when the speech/silence detector is used in conjunction with direct audio sources such as **Mic-in** and **Line-in**. Although this calibration is only used to set the parameters of the speech detector (assuming it is being used), the values displayed are useful for checking that both the audio equipment and computer are set-up correctly.

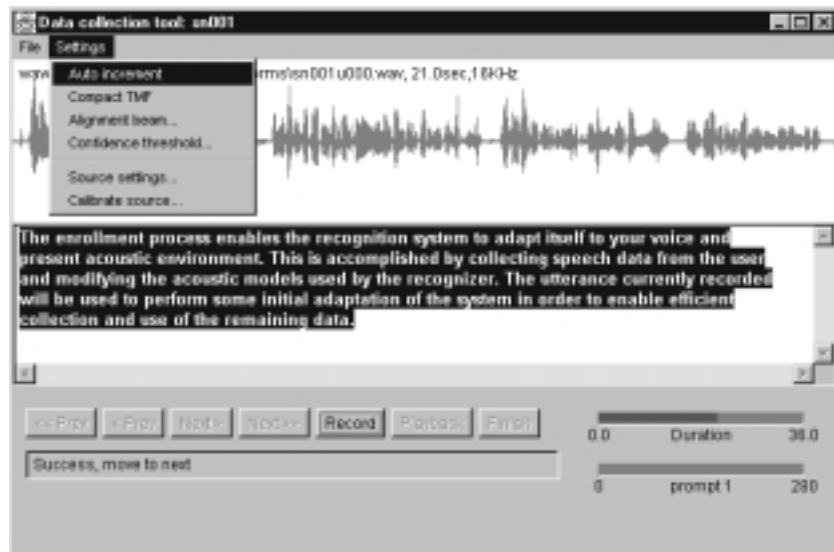


Fig. 5.1 The Enrollment Tool's main window.

5.1.2 The waveform panel

The waveform panel occupies the top part of the window and is used to display the recorded waveform data for the current utterance. During recordings, ensure that the microphone is positioned correctly and the input gain is set such that the speech signal is not clipped. The top left corner of the waveform panel, contains information such as waveform filename, overall duration and sampling frequency.

5.1.3 The Prompt area

The prompt area is used to display the current prompt.

5.1.4 The Controls

The control panel consists of the following buttons -

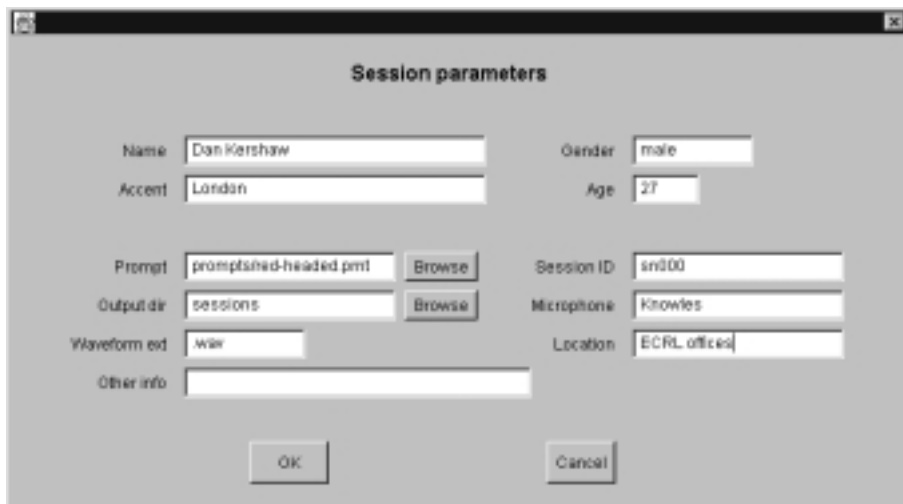
- **« Prev** - move backwards in the prompt set by 10 utterances
- **< Prev** - move to the previous prompt in the set
- **Next >** - move to the next prompt in the set
- **Next »** - move forward in the prompt set by 10 utterances
- **Record** - start recording. Once recording is in progress the button label will change to **Stop**. Click the button again to stop recording.
- **Playback** - replay the current utterance
- **Finish** - terminate the current enrollment session. The session information is saved and a generic adaptation transform is computed from the collected speech data. Optionally, a refined transform can be computed by adapting the acoustic model and calculating a new (improved) transform from the data. In all cases, the computation of refined transforms is strongly advised.

5.1.5 The Status area

The status area consists of two progress bars and a text field. The first progress bar (displayed in red) is active during recording and shows the duration of the recording so far (the initial range is 0-12 secs, however this changes as the current limit is reached). The second progress bar (displayed in blue) indicates how many prompts have been recorded so far with respect to the total number of prompts in the prompt file. The bottom left corner of the screen is used to display text status information.

5.2 Collecting adaptation data

Start the Enrollment tool from the **New enrollment** option in the **Adaptation ↓** menu of the system evaluation tool. The main window and the session configuration dialog (figure 5.2) will be displayed. Fill-in the requested user details such as name, gender, age, location, microphone used, etc. This will allow you to accurately identify the generated adaptation transform in later trials. Change the session ID to a meaningful identifier, for example “jon01”.



The image shows a 'Session parameters' dialog box with the following fields and values:

Field	Value
Name	Dan Kershaw
Gender	male
Accent	London
Age	27
Prompt	promptshd-headed.pml
Session ID	sn000
Output dir	sessions
Microphone	Knowles
Waveform ext	wav
Location	ECRL offices
Other info	

Buttons: OK, Cancel

Fig. 5.2 Session configuration dialog.

The data collection procedure is governed by a prompt file. The file contains prompt texts (one per line) for the speech utterances that the user will be asked to speak. The TRANSCRIBER distribution includes a default prompt file adapted from the Sherlock Holmes story “The Red-Headed League” by A. Conan Doyle. User defined prompts can be loaded and used by specifying their filename in the “Prompt” entry field.

Once the relevant information has been entered, click **OK** to dismiss the dialog. This will create a new session directory (jon01) in the specified location (default location is **sessions**) and load the specified prompt file. Once the prompts have been loaded, the first utterance will appear in the prompt area. Before recording, verify the current source settings by choosing **Source settings** from the **Settings** menu. Ensure the appropriate speech source is selected and the speech/silence detector is switched on. Click **OK** when finished.

Calibrate the speech/silence detector by choosing **Calibrate source** from the **Settings** menu. For more information on the calibration procedure refer to section 4.4. In a single-user environment with relatively stable background noise conditions calibration is only necessary when the enrollment tool is used for the first time. For the purpose of this example, disable the “auto-increment” mode from the **Settings** menu. The enrollment tool is now ready for collecting adaptation data.

To record a prompt click **Record** and read out the the current prompt. As the recording proceeds, the utterance text will be highlighted in sync with the read speech. The recording will be terminated automatically once silence is detected at the end of the utterance. If the prompt has been successfully recorded the message “Success, move to next” will be displayed in the status area and the waveform data will be shown in waveform panel. Information about the overall utterance duration, sampling frequency and waveform filename is shown in the top-left corner of the waveform panel (see figure 5.1). Click **Replay** to verify the utterance. If the utterance has not been recorded correctly verify the recording source and input levels (use the *Volume Control* accessory in WINDOWS or the relevant audio control panel in other operating systems). Repeat the recording if necessary. Once the utterance

has been recorded correctly, advance to the next prompt with the **Next >** button.

The enrollment procedure employs two mechanisms for judging if the currently recorded utterance is suitable for adaptation purposes. The first mechanism relies on an alignment procedure which matches the incoming speech against the text prompt. By the time the user has finished reading the prompt, the displayed text should be fully highlighted. If the system is unable to align the incoming speech to the text, the recording will be terminated and the message “Fail, repeat recording” will be displayed. If repeated failures are experienced, the user is advised to raise the alignment beam using the **Alignment Beam** option from the **Settings ↓** menu. Once the recorded utterance has been successfully aligned against the text prompt, the secondary rejection mechanism is invoked. This scheme uses the confidence scores computed for each of the aligned words in the utterance. If the smoothed confidence score value for any word in the utterance is less than the specified confidence threshold, the utterance is rejected. The confidence threshold can be set from the **Confidence threshold** option from the **Settings ↓** menu. Setting this value to 0.0 disables the confidence score rejection mechanism.

By default the enrollment tool operates in “auto-increment” mode. In this mode, once a prompt is successfully recorded, the system automatically advances to the next one and starts the recording (equivalent to clicking **Next**, **Record**).

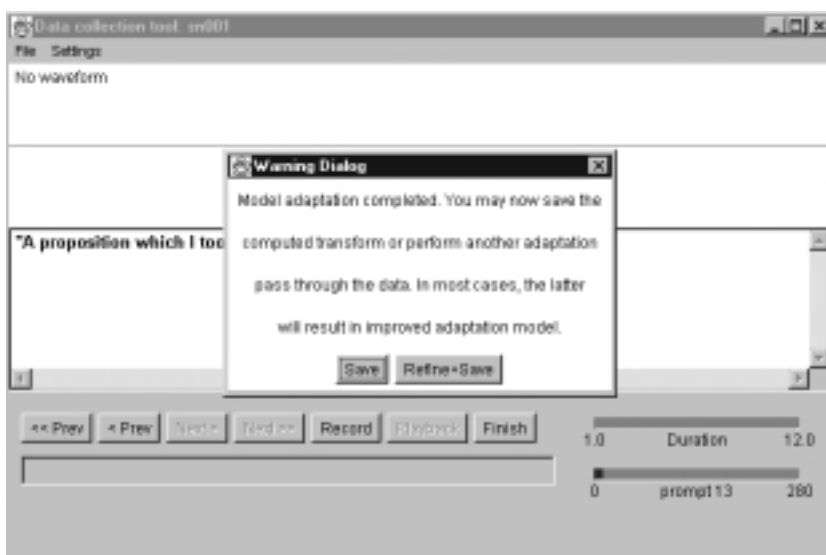


Fig. 5.3 Finishing an enrollment session.

Once a minimum of 20 prompts have been recorded, the enrollment session can be terminated by clicking **Finish**. The session information will be saved in the specified sessions directory and a generic adaptation transform will be computed from the collected speech data. At this point the user may choose to save the transform and exit (see figure 5.3). Alternatively, a refined transform can be computed by first adapting the acoustic model and then calculating a new transform from the data. The collected waveform data will be stored in the `jon01/waveforms` directory and it can be used to adapt other model sets.

5.3 Resuming enrollment

To resume an existing enrollment session, start the Enrollment tool using the **Resume enrollment** option from the **Adaptation** ↓ menu in the System evaluation tool. The Enrollment tool and the session file dialog will appear. Locate the desired session file (*.inf) and click **Open**. The session information and prompts will be loaded. If a session-specific transform model file (TMF) exists in the session directory (<session-id>.tmf) it will be automatically loaded and applied to the model set. If a TMF is not present, the enrollment tool will compute a generic transform from the adaptation data in the session. This may take several minutes depending on the amount of recorded data available.

This mechanism can also be used to compute an adaptation transform for other model sets using previously recorded data. In particular, any previously generated session-specific transforms should be renamed or moved to a different location. Then, the System evaluation tool can be started and a different system configuration chosen. An adaptation transform for the new model set will be generated by loading the enrollment session which contains the recorded adaptation data.

Chapter 6

HTK and HAPI based tools

The first part of this chapter describes the HAPI tools provided in the TRANSCRIBER distribution. HAPIADAPT enables the user to compute adaptation transforms and/or speaker dependent model sets from collected speech data. HAPIVITE is an example command-line speech recognition application. Both tools are also available as C source files in the HAPI/tutorial directory of the TRANSCRIBER distribution.

The second part of this chapter gives a brief description and the command line options of the HTK tools in the TRANSCRIBER distribution. HTK is a toolkit for building Hidden Markov Models (HMMs). HTK is not part of the TRANSCRIBER package and is available as a separate product from Entropic. However, for existing HTK customers, TRANSCRIBER provides replacements for some of the HTK tools with enhanced capabilities. In particular, HCopy and HList now include support for EPC data coding. User-generated HMM sets can be initialised for adaptation using HHED. Furthermore, HVITE can now optionally load an adaptation transform and apply it to the HMM set prior to performing recognition.

6.1 HAPIAdapt

6.1.1 Function

HAPIADAPT is a general-purpose offline adaptation tool, that can run in both supervised and unsupervised mode. After adaptation the HAPIADAPT tool produces two files :-

- The first file produced is a transform model file (TMF). This TMF contains the class-based transforms that can be applied to the HMM set MMF that the transforms were calculated for, in order to bring the HMM set closer to the target speaker and/or environment, typified by the enrollment/adaptation data.
- The second file produced is a master model file HMM set (MMF). This MMF contains an HMM set which has been adapted using a class+full transformation, and is to all intents and purposes a speaker-dependent model.

HAPIADAPT also features a very simple data collection mode, where the user is prompted to speak, and the speech waveforms stored in a specified directory (together with a script file). Collected data can then be used for adaptation purposes.

6.1.2 Use

HAPIADAPT is invoked via the command line

```
HAPIAdapt [options] DataFiles
```

The detailed operation of HAPIADAPT is controlled by HAPI configuration parameters, some of which can be overridden by command-line options.

- | | |
|----------------|--|
| -b s | Use a background/garbage word. The optional argument sets the background word name to s . When using a background model, if a word in the supervisory prompts is not found in the dictionary, then the background model is used. |
| -c dir | Performs a simple data collection mode only. This option disables any adaptation facilities that might be set with the other options. The file specified with the -p option is used to prompt the user. The speech waveforms are stored in the directory dir , together with a script file (listing the speech waveforms collected) called adaptation.scp . |
| -d dict | This specifies the main dictionary, dict , for the adaptation process. |
| -l list | Specify an HMM set list file. |
| -m | Use the MVX (or HTK) decoder only when performing unsupervised adaptation. This option is only applicable if an LVX decoder is available, in which case the LVX decoder is the default. |
| -p pmt | Load a file pmt containing all the prompts for the adaptation data. The prompt file must contain a single transcription per line, which is associated with each adaptation data file (or each line in the |

data script file). The transcription should not contain any punctuation, and must contain entries that can be found in the main dictionary (i.e. case and spelling must be correct). Out of vocabulary words can be automatically handled by the background/garbage model.

- s dict This specifies a secondary user dictionary, `dict`. This dictionary contains a pronunciation for the background model word, plus any other words the user may want to include for this application.
- t Save a compact format TMF only. Further updates (continued adaptation) is not possible with the compact TMF.
- u s Set the user id to `s`. This saves the user id in the TMF, names the TMF "`s.tmf`", and the MMF "`mmf_s.mmf`".
- w s Load in a recognition network `s`. If this is an LVX task, `s` is expected to be a language model, otherwise `s` is a recognition lattice. This is only loaded and used when a prompts file is not specified and unsupervised adaptation takes place. In this case all the words in the language model must be contained in the dictionary.
- A Echo the command-line.
- B Output HMM definition files (MMFs), and transform model files (TMFs) in binary format. Note that if the input MMF is binary encrypted (as per the supplied models), then the output is also binary encrypted.
- C config Load the configuration file `config`.
- H mmf Load HMM macro model file `mmf`. This option may be repeated to load multiple MMFs.
- J tmf Load in a transform model file `tmf`, and apply it to the model set.
- M s Save the TMF and MMF in the directory specified by `s`. If this option is not specified then the files are saved in the current directory.
- S s Load in the script file `s`. The script file contains a list of speech data files to be used for the offline adaptation process.

6.1.3 Tracing

HAPIADAPT supports the following trace options where each trace flag is given using an octal base

- 0001 enable basic progress reporting.

Trace flags are set using the `-T` option or the `TRACE` configuration variable.

6.2 HAPIVite

6.2.1 Function

HAPIVITE is a general-purpose Viterbi word recogniser, which mirrors the HTK tool HVITE. However, HAPIVITE uses calls solely to the HTK API, HAPI. It will match a speech file against a network of HMMs and output a transcription for each. When performing N-best recognition a word level lattice containing multiple hypotheses can also be produced.

A word level lattice file or network file is read in and then expanded using the supplied dictionary to create a model based network. This allows arbitrary finite state word networks and simple forced alignment to be specified.

This expansion can be used to create context independent, word internal context dependent and cross word context dependent networks. The way in which the expansion is performed is determined automatically from the dictionary and HMMList. When all labels appearing in the dictionary are defined in the HMMList no expansion of model names is performed. Otherwise if all the labels in the dictionary can be satisfied by models dependent only upon word internal context these will be used else cross word context expansion will be performed. These defaults can be overridden by HNET configuration parameters. Other default files and settings can be overwritten by HAPI configuration parameters (see the HAPI Book).

HAPIVITE supports shared parameters and appropriately pre-computes output probabilities. For increased processing speed, HAPIVITE can optionally perform a beam search controlled by a user specified threshold (see `-t` option). When fully tied mixture models are used, observation pruning is also provided (see the `-c` option).

6.2.2 Use

HAPIVITE is invoked via the command line

```
HAPIVite [options] dictFile hmmList DataFiles ...
```

HAPIVITE will either load a single network file and match this against each of the test files `-w netFile`, or create a new network for each test file from a word lattice `-w`. When a new network is created for each test file the path name of the lattice file to load is determined from the test file name and the `-L` and `-X` options described below.

If the `-W` option is specified, then a (large vocabulary) recognition task using a single language model is assumed. The `-w s` option is used to specify the language model `s`, which is converted into a network and this is then matched against each test file or incoming speech.

If no `testFiles` are specified the `-w s` option must be specified and recognition will be performed direct from audio. When recognising directly from audio, a speech/silence calibration is performed before recognition begins. Recognition output is sent to standard output and can be saved to file using the `-e` option.

The `hmmList` should contain a list of the models required to construct the network from the word level representation.

HAPIVITE can also load a transform model file (TMF). The transforms stored in the TMF are applied to the acoustic model set before the recognition process starts. Incremental online adaptation is also available, either starting from scratch with the

input acoustic model, or from a loaded TMF (containing statistics). At the end of the session the updated TMF can be saved for later use.

The recogniser output is written in the form of a label file whose path name is determined from the test file name and the `-l` and `-x` options described below. The list of test files can be stored in a script file if required.

When performing N-best recognition (see `-n` *N* option described below) the output label file can contain multiple alternatives `-n` *N* *M* and a lattice file containing multiple hypotheses can be produced.

The detailed operation of HAPIVITE is controlled by the following command line options

- `-a` Perform alignment. HAPIVITE will load a lattice file and create an alignment network for each test file from the lattice.
- `-b s` Use *s* as the sentence boundary during alignment.
- `-c f` Set the tied-mixture observation pruning threshold to *f*. When all mixtures of all models are tied to create a full tied-mixture system, the calculation of output probabilities is treated as a special case. Only those mixture component probabilities which fall within *f* of the maximum mixture component probability are used in calculating the state output probabilities (default 10.0).
- `-d dir` This specifies the directory to search for the HMM definition files corresponding to the labels used in the recognition network.
- `-e` When using direct audio input, output transcriptions are not normally saved. When this option is set, each output transcription is written to a file called *PnS* where *n* is an integer which increments with each output file, *P* and *S* are strings which are by default empty but can be set using the configuration variables `RECOUTPREFIX` and `RECOUTSUFFIX`.
- `-f` During recognition keep track of full state alignment.
- `-g` When using direct audio input, this option enables audio replay of each input utterance after it has been recognised.
- `-i s` Output transcriptions to MLF *s*.
- `-j i` Perform incremental online adaptation, either from scratch or from a TMF (containing statistics). The transform is updated and applied to the acoustic model set every *i* utterances (default 1).
- `-k i` Compute the the `OutP` forwards for *i* frames.
- `-l dir` This specifies the directory to store the output label files. If this option is not used then HAPIVITE will store the label files in the same directory as the data. When output is directed to an MLF, this option can be used to add a path to each output file name. In particular, setting the option `-l '*'` will cause a label file named *xxx* to be prefixed by the pattern `"/xxx"` in the output MLF file. This is useful for generating MLFs which are independent of the location of the corresponding data files.

- m During recognition keep track of model boundaries.
- n i [N] Use i tokens in each state to perform N-best recognition. The number of alternative output hypotheses N defaults to 1.
- o s Choose how the output labels should be formatted. s is a string with certain letters (from **NSCTWMX**) indicating binary flags that control formatting options. N normalise acoustic scores by dividing by the duration (in frames) of the segment. S remove scores from output label. By default scores will be set to the total likelihood of the segment. C Set the transcription labels to start and end on frame centres. By default start times are set to the start time of the frame and end times are set to the end time of the frame. T Do not include times in output label files. W Do not include words in output label files when performing state or model alignment. X Strip triphones down to monophones. M Do not include model names in output label files when performing state and model alignment.
- p f Set the word insertion log probability to f (default 0.0).
- q s Choose how the output lattice should be formatted. s is a string with certain letters (from **ABtvaldmn**) indicating binary flags that control formatting options. A attach word labels to arcs rather than nodes. B output lattices in binary for speed. t output node times. v output pronunciation information. a output acoustic likelihoods. l output language model likelihoods. d output word alignments (if available). m output within word alignment durations. n output within word alignment likelihoods.
- r f Set the dictionary pronunciation probability scale factor to f. (default value 1.0).
- s f Set the grammar scale factor to f. This factor post-multiplies the language model likelihoods from the word lattices. (default value 1.0).
- t f Enable beam searching such that any model whose maximum log probability token falls more than f below the maximum for all models is deactivated. Setting f to 0.0 disables the beam search mechanism (default value 0.0).
- u i Set the maximum number of active models to i. Setting i to 0 disables this limit (default 0).
- v f Enable word end pruning. Do not propagate tokens from word end nodes that fall more than f below the maximum word end likelihood. (default 0.0).
- w [s] Perform recognition from word level networks. If s is included then use it to define the network used for every file.
- x ext This sets the extension to use for HMM definition files to ext.

- `-y ext` This sets the extension for output label files to `ext` (default `rec`).
- `-z ext` Enable output of lattices (if performing NBest recognition) with extension `ext` (default off).
- `-J tmf` Load the transform model file `tmf`.
- `-K tmf` Save an updated transform set including statistics in the transform model file `tmf`.
- `-L dir` This specifies the directory to find input label (when `-a` is specified) or network files (when `-w` is specified).
- `-W` Do an LVR decode using a language model for building the network. If this is not set HAPIVITE defaults to using a network or lattice to build the network. (For alignment a new network is created from a lattice for each input file.)
- `-X s` Set the extension for the input label or network files to be `s` (default value `lab`).
- `-F fmt` Set the source data format to `fmt`.
- `-G fmt` Set the label file format to `fmt`.
- `-H mmf` Load HMM macro model file `mmf`. This option may be repeated to load multiple MMFs.
- `-P fmt` Set the target label format to `fmt`.

HAPIVITE also supports the standard options `-A`, `-C`, `-D`, `-S`, `-T`, and `-V` as described in section 8.4.

Tracing

HAPIVITE supports the following trace options where each trace flag is given using an octal base

- 0001 enable basic progress reporting.
- 0002 list observations.
- 0004 frame-by-frame best token.
- 0010 show memory usage at start and finish.
- 0020 show memory usage after each utterance.

Trace flags are set using the `-T` option or the `TRACE` configuration variable.

6.3 HCopy

6.3.1 Function

This program will copy one or more data files to a designated output file, optionally converting the data into a parameterised form. While the source files can be in any supported format, the output format is always HTK. By default, the whole of the source file is copied to the target but options exist to only copy a specified segment. Hence, this program is used to convert data files in other formats to the HTK format, to concatenate or segment data files, and to parameterise the result. If any option is set which leads to the extraction of a segment of the source file rather than all of it, then segments will be extracted from all source files and concatenated to the target.

Labels will be copied/concatenated if any of the options indicating labels are specified (`-i -l -x -G -I -L -P -X`). In this case, each source data file must have an associated label file, and a target label file is created. The name of the target label file is the root name of the target data file with the extension `.lab`, unless the `-X` option is used. This new label file will contain the appropriately copied/truncated/concatenated labels to correspond with the target data file; all start and end boundaries are recalculated if necessary.

When used in conjunction with HSLAB, HCopy provides a facility for tasks such as cropping silence surrounding recorded utterances. Since input files may be coerced, HCopy can also be used to convert the parameter kind of a file, for example from WAVEFORM to MFCC, depending on the configuration options. Not all possible conversions can actually be performed; see Table 6.1 for a list of valid conversions. Conversions must be specified via a configuration file as described in the HTK Book.

6.3.2 Use

HCOPY is invoked by typing the command line

```
HCopy [options] sa1 [ + sa2 + ... ] ta [ sb1 [ + sb2 + ... ] tb ... ]
```

This causes the contents of the one or more source files `sa1`, `sa2`, ... to be concatenated and the result copied to the given target file `ta`. To avoid the overhead of reinvoking the tool when processing large databases, multiple sources and targets may be specified, for example

```
HCopy srcA.wav + srcB.wav tgtAB.wav srcC.wav tgtD.wav
```

will create two new files `tgtAB.wav` and `tgtD.wav`. HCopy takes file arguments from a script specified using the `-S` option exactly as from the command line, except that any newlines are ignored.

The allowable options to HCopy are as follows where all times and durations are given in 100 ns units and are written as floating-point numbers.

- `-a i` Use level `i` of associated label files with the `-n` and `-x` options. Note that this is not the same as using the `TRANSLEVEL` configuration variable since the `-a` option still allows all levels to be copied through to the output files.
- `-e f` End copying from the source file at time `f`. The default is the end of the file. If `f` is negative or zero, it is interpreted as a time

relative to the end of the file, while a positive value indicates an absolute time from the start of the file.

- `-i mlf` Output label files to master file `mlf`.
- `-l s` Output label files to the directory `s`. The default is to output to the current directory.
- `-m t` Set a margin of duration `t` around the segments defined by the `-n` and `-x` options.
- `-n i [j]` Extract the speech segment corresponding to the `i`'th label in the source file. If `j` is specified, then the segment corresponding to the sequence of labels `i` to `j` is extracted. Labels are numbered from their position in the label file. A negative index can be used to count from the end of the label list. Thus, `-n 1 -1` would specify the segment starting at the first label and ending at the last.
- `-s f` Start copying from the source file at time `f`. The default is 0.0, ie the beginning of the file.
- `-t n` Set the line width to `n` chars when formatting trace output.
- `-x s [n]` Extract the speech segment corresponding to the first occurrence of label `s` in the source file. If `n` is specified, then the `n`'th occurrence is extracted. If multiple files are being concatenated, segments are extracted from each file in turn, and the label must exist for each concatenated file.
- `-F fmt` Set the source data format to `fmt`.
- `-G fmt` Set the label file format to `fmt`.
- `-I mlf` This loads the master label file `mlf`. This option may be repeated to load several MLFs.
- `-L dir` Search directory `dir` for label files (default is to search current directory).
- `-O fmt` Set the target data format to `fmt`.
- `-P fmt` Set the target label format to `fmt`.
- `-X ext` Set label file extension to `ext` (default is `lab`).

HCOPY also supports the standard options `-A`, `-C`, `-D`, `-S`, `-T`, and `-V` as described in section 8.4.

Note that the parameter kind conversion mechanisms described in the HTK Book will be applied to all source files. In particular, if an automatic conversion is requested via the configuration file, then HCOPY will copy or concatenate the converted source files, not the actual contents. Similarly, automatic byte swapping may occur depending on the source format and the configuration variable `BYTEORDER`. Because the sampling rate may change during conversions, the options that specify a position within a file i.e. `-s` and `-e` use absolute times rather than sample index numbers. All times in HTK are given in units of 100ns and are written as floating-point numbers. To save writing long strings of zeros, standard exponential notation may be used, for example `-s 1E6` indicates a start time of 0.1 seconds from the beginning of the file.

	Outputs									
	W	A	V	E	F	O	R	M		
Inputs	L	P	C	E	I	R	F	M	B	S
	P	P	R	S	E	T	E	F	A	P
	C	C	A	C	C	C	K	C	R	C
WAVEFORM	✓	✓	✓	✓	✓	✓	✓	✓		✓
LPC		✓	✓	✓	✓					✓
LPREFC		✓	✓	✓	✓					✓
LPCEPSTRA		✓	✓	✓	✓					✓
IREFC		✓	✓	✓	✓					✓
MFCC							✓			✓
FBANK							✓	✓		✓
MELSPEC							✓	✓	✓	✓
USER									✓	✓
EPC										✓
DISCRETE										✓

Table. 6.1 Valid Parameter Conversions

Note that truncations are performed *after* any desired coding, which may result in a loss of time resolution if the target file format has a lower sampling rate. Also, because of windowing effects, truncation, coding, and concatenation operations are not necessarily interchangeable. If in doubt, perform all truncation/concatenation in the waveform domain and then perform parameterisation as a last, separate invocation of HCopy.

6.3.3 Trace Output

HCOPY supports the following trace options where each trace flag is given using an octal base

- 00001 basic progress reporting.
- 00002 source and target file formats and parameter kinds.
- 00004 segment boundaries computed from label files.
- 00010 display memory usage after processing each file.

Trace flags are set using the -T option or the TRACE configuration variable.

6.4 HHEd

6.4.1 Function

HHEd is a script driven editor for manipulating sets of HMM definitions. Its basic operation is to load in a set of HMMs, apply a sequence of edit operations and then output the transformed set. HHEd is mainly used for applying tyings across selected HMM parameters. It also has facilities for cloning HMMs, clustering states and editing HMM structures.

Many HHEd commands operate on sets of similar items selected from the set of currently loaded HMMs. For example, it is possible to define a set of all final states of all vowel models, or all mean vectors of all mixture components within the model X, etc. Sets such as these are defined by item lists using the syntax rules given below. In all commands, all of the items in the set defined by an item list must be of the same type where the possible types are

s	– state	t	– transition matrix
p	– pdf	w	– stream weights
m	– mixture component	d	– duration parameters
u	– mean vector	x	– transform matrix
v	– variance vector	i	– inverse covariance matrix
h	– HMM definition		

Most of the above correspond directly to the tie points shown in Fig 7.8 of the HTK Book. There is just one exception. The type “p” corresponds to a pdf (ie a sum of Gaussian mixtures). Pdf’s cannot be tied, however, they can be named in a Tie (TI) command (see below) in which case, the effect is to join all of the contained mixture components into one pool of mixtures and then all of the mixtures in the pool are shared across all pdf’s. This allows conventional *tied-mixture* or *semi-continuous* HMM systems to be constructed.

The syntax rules for item lists are as follows. An item list consists of a comma separated list of item sets.

```
itemList      =  "{" itemSet { "," itemSet } "}"
```

Each `itemSet` consists of the name of one or more HMMs (or a pattern representing a set of HMMs) followed by a specification which represents a set of paths down the parameter hierarchy each terminating at one of the required parameter items.

```
itemSet       =  hmmName . [ "transP" | "state" state ]
hmmName       =  ident | identList
identList     =  "(" ident { "," ident } ")"
ident         =  < char | metachar >
metachar      =  "?" | "★"
```

A `hmmName` consists of a single `ident` or a comma separated list of `ident`’s. The following examples are all valid `hmmName`’s:

```
aa three model001 (aa,iy,ah,u) (one,two,three)
```

In addition, an `ident` can contain the metacharacter “?” which matches any single character and the metacharacter “★” which matches a string of zero or more characters. For example, the item list

```
{*-aa+*.transP}
```

would represent the set of transition matrices of all loaded triphone variations of **aa**.

Items within states require the state indices to be specified

```
state      = index [ "." stateComp ]
index      = "[" intRange { "." intRange } "]"
intRange   = integer [ "-" integer ]
```

For example, the item list

```
{*.state[1,3-5,9]}
```

represents the set of all states 1, 3 to 5 inclusive and 9 of all currently loaded HMMs. Items within states include durational parameters, stream weights, pdf's and all items within mixtures

```
stateComp  = "dur" | "weights" | [ "stream" index ] "." "mix" [ mix ]
```

For example,

```
{(aa,ah,ax).state[2].dur}
```

denotes the set of durational parameter vectors from state 2 of the HMMs **aa**, **ah** and **ax**. Similarly,

```
{*.state[2-4].weights}
```

denotes the set of stream weights for states 2 to 4 of all currently loaded HMMs. The specification of pdf's may optionally include a list of the relevant streams, if omitted, stream 1 is assumed. For example,

```
{three.state[3].mix}
```

and

```
{three.state[3].stream[1].mix}
```

both denote a list of the single pdf belonging to stream 1 of state 3 of the HMM **three**.

Within a pdf, the possible item types are mixture components, mean vectors, and the various possible forms of covariance parameters

```
mix        = index [ "." ( "mean" | "cov" ) ]
```

For example,

```
{*.state[2].mix[1-3]}
```

denotes the set of mixture components 1 to 3 from state 2 of all currently loaded HMMs and

```
{(one,two).state[4].stream[3].mix[1].mean}
```

denotes the set of mean vectors from mixture component 1, stream 3, state 4 of the HMMs **one** and **two**. When **cov** is specified, the type of the covariance item referred to is determined from the **CovKind** of the loaded models. Thus, for diagonal covariance models, the item list

```
{*.state[2-4].mix[1].cov}
```

would denote the set of variance vectors for mixture 1, states 2 to 4 of all loaded HMMs.

Note finally, that it is not an error to specify non-existent models, states, mixtures, etc. All item list specifications are regarded as patterns which are matched against the currently loaded set of models. All and only those items which match are included in the set. However, both a null result and a set of items of mixed type do result in errors.

All HHEd commands consist of a 2 character command name followed by zero or more arguments. In the following descriptions, item lists are shown as `itemList(c)` where the character `c` denotes the type of item expected by that command. If this type indicator is missing then the command works for all item types.

The HHEd commands are as follows

AT i j prob itemList(t)

Add a transition from state `i` to state `j` with probability `prob` for all transition matrices in `itemList`. The remaining transitions out of state `i` are rescaled so that $\sum_k a_{ik} = 1$. For example,

```
AT 1 3 0.1 {*.transP}
```

would add a skip transition to all loaded models from state 1 to state 3 with probability 0.1.

AU hmmList

Use a set of decision trees to create a new set of models specified by the `hmmList`. The decision trees may be made as a result of either the `TB` or `LT` command.

Each model in `hmmList` is constructed in the following manner. If a model with the same logical name already exists in the current HMM set this is used unchanged, otherwise the model is synthesised from the decision trees. If the trees cluster at the model level the synthesis results in a logical model sharing the physical model from the tree that matches the new context. If the clustering was performed at the state level a prototype model (an example of the same phone model occurring in a different context) is found and a new HMM is constructed that shares the transition matrix with the prototype model but consists of tied states selected using the decision tree.

CL hmmList

Clone a HMM list. The file `hmmList` should hold a list of HMMs all of whose logical names are either the same as, or are context-dependent versions of the currently loaded set of HMMs. For each name in `hmmList`, the corresponding HMM in the loaded set is cloned. On completion, the currently loaded set is discarded and replaced by the new set. For example, if the file `mylist` contained

```
A-A+A
A-A+B
B-A+A
B-B+B
B-B+A
```

and the currently loaded HMMs were just A and B, then A would be cloned 3 times to give the models A-A+A, A-A+B and B-A+A, and B would be cloned 2 times to give B-B+B and B-B+A. On completion, the original definitions for A and B would be deleted (they could be retained by including them in the new `hmmList`).

CO newList

Compact a set of HMMs. The effect of this command is to scan the currently loaded set of HMMs and identify all identical definitions. The physical name of the first model in each identical set is then assigned to all models in that set and all model definitions are replaced by a pointer to the first model definition. On completion, a new list of HMMs which includes the new model tyings is written out to file `newList`. For example, suppose that models A, B, C and D were currently loaded and A and B were identical. Then the command

```
CO tlist
```

would tie HMMs A and B, set the physical name of B to A and output the new HMM list

```
A
B A
C
D
```

to the file `tlist`. This command is used mainly after performing a sequence of parameter tying commands.

DP s n id ...

Duplicates a set of HMMs. This command is used to replicate a set of HMMs whilst allowing control over which structures will be shared between them. The first parameter controls duplication of tied structures. Any macros whose type appears in string `s` are duplicated with new names and only used in the duplicate model set. The remaining shared structures are common through all the model sets (original and duplicates). The second parameter defines the number of times the current HMM set should be duplicated with the remaining `n` parameters providing suffices to make the original macro identifiers unique in each duplicated HMM set.

For instance the following script could be used to duplicate a set of tied state models to produce gender dependent ones with tied variances.

```
MM "v_" { (*).state[2-4].mix[1-2].cov }
DP "v" 2 ":m" ":f"
```

The MM command converts all variances into macros (with each macro referring to only one variance). The DP command then duplicates the current HMM set twice. Each of the duplicate sets will share the tied variances with the original set but will have new mixture means, weights and state macros. The new macro names will be constructed by appending the id `":m"` or `":f"` to the original macro name whilst the model names have the id appended after the base phone name (so `ax-b+d` becomes `ax-b:m+d` or `ax-b:f+d`).

J0 size minw

Set the size and minimum mixture weight for subsequent Tie (TI) commands applied to pdf's. The value of **size** sets the total number of mixtures in the tied mixture set (*codebook*) and **minw** sets a floor on the mixture weights as a multiple of MINMIX. This command only applies to tying item lists of type "p" (see the Tie TI command below).

LS statsfile

This command is used to read in the HEREST statistics file (see the HEREST -s option) stored in **statsfile**. These statistics are needed for certain clustering operations. The statistics file contains the occupation count for every HMM state.

LT treesfile

This command reads in the decision trees stored in **treesfile**. The trees file will consist of a set of questions defining contexts that may appear in the subsequent trees. The trees are used to identify either the state or the model that should be used in a particular context. The file would normally be produced by ST after tree based clustering has been performed.

MM macro itemList

This command makes each item (I=1..N) in **itemList** into a macro with name **nameI** and a usage of one. This command can prevent unnecessary duplication of structures when HMMs are cloned or duplicated.

MT triList newTriList

Make a set of triphones by merging the currently loaded set of biphones. This is a very specialised command. All currently loaded HMMs must have 3 emitting states and be either left or right context-dependent biphones. The list of HMMs stored in **triList** should contain one or more triphones. For each triphone in **triList** of the form **X-Y+Z**, there must be currently loaded biphones **X-Y** and **Y+Z**. A new triphone **X-Y+Z** is then synthesised by first cloning **Y+Z** and then replacing the state information for the initial emitting state by the state information for the initial emitting state of **X-Y**. Note that the underlying physical names of the biphones used to create the triphones are recorded so that where possible, triphones generated from tied biphones are also tied. On completion, the new list of triphones including aliases is written to the file **newTriList**.

MU m itemList(p)

Increase the number of non-defunct mixture components in each pdf in the **itemList** to **m** (when **m** is just a number) or by **m** (when **m** is a number preceded by a + sign. A defunct mixture is one for which the weight has fallen below MINMIX. This command works in two steps. Firstly, the weight of each mixture in each pdf is checked. If any defunct mixtures are discovered, then each is successively replaced by a non-defunct mixture component until either the required total number of non-defunct mixtures is reached or there are no defunct mixtures left. This replacement works by first deleting the defunct mixture and then finding the mixture with the largest

weight and splitting it. The split operation is as follows. The weight of the mixture component is first halved and then the mixture is cloned. The two identical mean vectors are then perturbed by adding 0.2 standard deviations to one and subtracting the same amount from the other.

In the second step, the mixture component with the largest weight is split as above. This is repeated until the required number of mixture components are obtained. Whenever, a mixture is split, a count is incremented for that mixture so that splitting occurs evenly across the mixtures. Furthermore, a mixture whose *gconst* value falls more than four standard deviations below the mean is not split.

As an example, the command

```
MU 6 {-aa*.state[3].mix}
```

would increase the number of mixture components in state 3 of all triphones of **aa** to 6.

NC N macro itemList(s)

N-cluster the states listed in the `itemList` and tie each cluster *i* as macro `macroi` where *i* is 1,2,3,...,N. The set of states in the `itemList` are divided into N clusters using the following furthest neighbour hierarchical cluster algorithm:

```
create 1 cluster for each state;
n = number of clusters;
while (n>N) {
    find i and j for which g(i,j) is minimum;
    merge clusters i and j;
}
```

Here $g(i,j)$ is the inter-group distance between clusters *i* and *j* defined as the maximum distance between any state in cluster *i* and any state in cluster *j*. The calculation of the inter-state distance depends on the type of HMMs involved. Single mixture Gaussians use

$$d(i,j) = \frac{1}{S} \sum_{s=1}^S \left[\frac{1}{V_s} \sum_{k=1}^{V_s} \frac{(\mu_{isk} - \mu_{jsk})^2}{\sigma_{isk} \sigma_{jsk}} \right]^{\frac{1}{2}} \quad (6.1)$$

where V_s is the dimensionality of stream *s*. Fully tied mixture systems (ie **TIEDHS**) use

$$d(i,j) = \frac{1}{S} \sum_{s=1}^S \left[\frac{1}{M_s} \sum_{m=1}^{M_s} (c_{ism} - c_{jsm})^2 \right]^{\frac{1}{2}} \quad (6.2)$$

and all others use

$$d(i,j) = -\frac{1}{S} \sum_{s=1}^S \frac{1}{M_s} \sum_{m=1}^{M_s} \log[b_{js}(\mu_{ism})] + \log[b_{is}(\mu_{jsm})] \quad (6.3)$$

where $b_{js}(x)$ is as defined in equation 7.1 in the HTK Book for the continuous case and equation 7.2 in the HTK Book for the discrete case. The actual tying of the states in each cluster is performed exactly as for the Tie (TI) command below. The macro for the *i*'th tied cluster is called `macroi`.

QS name itemList(h)

Define a question **name** which is true for all the models in **itemList**. These questions can subsequently be used as part of the decision tree based clustering procedure (see TB command below).

RC N identifier [itemlist]

This command is used to grow a regression class tree for adaptation purposes. A regression class tree is grown with **N** terminal or leaf nodes, using the centroid splitting algorithm with a Euclidean distance measure to cluster the model set's mixture components. Hence each leaf node specifies a particular mixture component cluster. The regression class tree is saved with the macro identifier **identifier.N**. Each Gaussian component is also labelled with a regression class number (corresponding to the leaf node number that the Gaussian component resides in). In order to grow the regression class tree it is necessary to load in a **statsfile** using the LS command. It is also possible to specify an **itemlist** containing the "non-speech" sound components such as the silence mixture components. If this is included then the first split made will result in one leaf containing the specified non-speech sound components, while the other leaf will contain the rest of the model set components. Tree construction then continues as usual.

RN hmmIdName

Rename or add the hmm set identifier in the global options macro to **hmmIdName**.

RM hmmFile

Load the hmm from **hmmFile** and subtract the mean from state 2, mixture 1 of the model from every loaded model. Every component of the mean is subtracted including deltas and accelerations.

RO f [statsfile]

This command is used to remove outlier states during clustering with subsequent NC or TC commands. If **statsfile** is present it first reads in the HEREST statistics file (see LS) otherwise it expects a separate LS command to have already been used to read in the statistics. Any subsequent NC, TC or TB commands are extended to ensure that the occupancy clusters produced exceeds the threshold **f**. For TB this is used to choose which questions are allowed to be used to split each node. Whereas for NC and TC a final merging pass is used and for as long the smallest cluster count falls below the threshold **f**, then that cluster is merged with its nearest neighbour.

RT i j itemList(t)

Remove the transition from state **i** to **j** in all transition matrices given in the **itemList**. After removal, the remaining non-zero transition probabilities for state **i** are rescaled so that $\sum_k a_{ik} = 1$.

SH

Show the current HMM set. This command can be inserted into edit scripts for debugging. It prints a summary of each loaded HMM identifying any tied parameters.

SK skind

Change the sample kind of all loaded HMMs to **skind**. This command is typically used in conjunction with the **SW** command. For example, to add delta coefficients to a set of models, the **SW** command would be used to double the stream widths and then this command would be used to add the **_D** qualifier.

SS N

Split into **N** independent data streams. This command causes the currently loaded set of HMMs to be converted from 1 data stream to **N** independent data streams. The widths of each stream are determined from the single stream vector size and the sample kind as described in section 5.10 of the HTK Book. Execution of this command will cause any tyings associated with the split stream to be undone.

ST filename

Save the currently defined questions and trees to file **filename**. This allows subsequent construction of models using for new contexts using the **LT** and **AU** commands.

SU N w1 w2 w3 .. wN

Split into **N** independent data streams with stream widths as specified. This command is similar to the **SS** command except that the width of each stream is defined explicitly by the user rather than using the built-in stream splitting rules. Execution of this command will cause any tyings associated with the split stream to be undone.

SW s n

Change the width of stream **s** of all currently loaded HMMs to **n**. Changing the width of stream involves changing the dimensions of all mean and variance vectors or covariance matrices. If **n** is greater than the current width of stream **s**, then mean vectors are extended with zeroes and variance vectors are extended with 1's. Covariance matrices are extended with zeroes everywhere except for the diagonal elements which are set to 1. This command preserves any tyings which may be in force.

TB f macro itemList(s or h)

Decision tree cluster all states in the given **itemList** and tie them as **macro_i** where **i** is 1,2,3,... This command performs a top down clustering of the states or models appearing in **itemlist**. This clustering starts by placing all items in a single root node and then choosing a question from the current set to split the node in such a way as to maximise the likelihood of a single diagonal covariance Gaussian at each of the child nodes generating the training data. This splitting continues until the increase in likelihood falls below threshold **f** or no questions are available which do

not pass the outlier threshold test. This type of clustering is only implemented for single mixture, diagonal covariance untied models.

TC f macro itemList(s)

Cluster all states in the given **itemList** and tie them as **macro_i** where **i** is 1,2,3,... This command is identical to the **NC** command described above except that the number of clusters is varied such that the maximum within cluster distance is less than the value given by **f**.

TI macro itemList

Tie the items in **itemList** and assign them to the specified **macro** name. This command applies to any item type but all of the items in **itemList** must be of the same type. The detailed method of tying depends on the item type as follows:

state(s) the state with the largest total value of **gConst** in stream 1 (indicating broad variances) and the minimum number of defunct mixture weights (see **MU** command) is selected from the item list and all states are tied to this typical state.

transitions(t) all transition matrices in the item list are tied to the first in the list.

mixture(m) all mixture components in the item list are tied to the first in the list.

mean(u) the average vector of all the mean vectors in the item list is calculated and all the means are tied to this average vector.

variance(v) a vector is constructed for which each element is the maximum of the corresponding elements from the set of variance vectors to be tied. All of the variances are then tied to this maximum vector.

covariance(i) all covariance matrices in the item list are tied to the first in the list.

xform(x) all transform matrices in the item list are tied to the first in the list.

duration(d) all duration vectors in the item list are tied to the first in the list.

stream weights(w) all stream weight vectors in the item list are tied to the first in the list.

pdf(p) as noted earlier, pdf's are tied to create tied mixture sets rather than to create a shared pdf. The procedure for tying pdf's is as follows

1. All mixtures from all pdf's in the item list are collected together in order of mixture weight.
2. If the number of mixtures exceeds the join size J [see the **Join (JO)** command above], then all but the first J mixtures are discarded.
3. If the number of mixtures is less than J , then the mixture with the largest weight is repeatedly split until there are exactly J mixture components. The split procedure used is the same as for the **MixUp (MU)** command described above.

4. All pdf's in the item list are made to share all J mixture components. The weight for each mixture is set proportional to the log likelihood of the mean vector of that mixture with respect to the original pdf.
5. Finally, all mixture weights below the floor set by the Join command are raised to the floor value and all of the mixture weights are renormalised.

TR *n*

Change the level of detail for tracing and consists of a number of separate flags which can be added together. Values 0001, 0002, 0004, 0008 have the same meaning as the command line trace level but apply only to a single block of commands (a block consisting of a set of commands of the name). A value of 0010 can be used to show current memory usage.

UT *itemList*

Untie all items in *itemList*. For each item in the item list, if the usage counter for that item is greater than 1 then it is cloned, the original shared item is replaced by the cloned copy and the usage count of the shared item is reduced by 1. If the usage count is already 1, the associated macro is simply deleted and the usage count set to 0 to indicate an unshared item. Note that it is not possible to untie a pdf since these are not actually shared [see the Tie (TI) command above].

6.4.2 Use

HHED is invoked by typing the command line

```
HHed [options] edCmdFile hmmList
```

where *edCmdFile* is a text file containing a sequence of edit commands as described above and *hmmList* defines the set of HMMs to be edited (see HMODEL for the format of HMM list). If the models are to be kept in separate files rather than being stored in an MMF, the configuration variable `KEEPDISTINCT` should be set to true. The available options for HHED are

- d *dir* This option tells HHED to look in the directory *dir* to find the model definitions.
- o *ext* This causes the file name extensions of the original models (if any) to be replaced by *ext*.
- w *mmf* Save all the macros and model definitions in a single master macro file *mmf*.
- x *s* Set the extension for the edited output files to be *s* (default is to use the original names unchanged).
- z Setting this option causes all aliases in the loaded HMM set to be deleted (zapped) immediately before loading the definitions. The result is that all logical names are ignored and the actual HMM list consists of just the physically distinct HMMs.
- B Output HMM definition files in binary format.

- H *mmf* Load HMM macro model file *mmf*. This option may be repeated to load multiple MMFs.
- M *dir* Store output HMM macro model files in the directory *dir*. If this option is not given, the new HMM definition will overwrite the existing one.
- Q Print a summary of all commands supported by this tool.

HHED also supports the standard options -A, -C, -D, -S, -T, and -V as described in section 8.4.

6.4.3 Tracing

HHED supports the following trace options where each trace flag is given using an octal base

- 00001 basic progress reporting.
- 00002 intermediate progress reporting.
- 00004 detailed progress reporting.
- 00010 show item lists used for each command.
- 00020 show memory usage.
- 00100 show changes to macro definitions.
- 00200 show changes to stream widths.
- 00400 show clusters.
- 00800 show questions.
- 01000 show tree filtering.
- 02000 show tree splitting.
- 04000 show tree merging.
- 10000 show good question scores.
- 20000 show all question scores.
- 40000 show all merge scores.

Trace flags are set using the -T option or the TRACE configuration variable.

6.5 HList

6.5.1 Function

This program will list the contents of one or more data sources in any HTK supported format. It uses the full HTK speech input facilities described in chapter 5 of the HTK Book and it can thus read data from a waveform file, from a parameter file and direct from an audio source. HLIST provides a dual rôle in HTK. Firstly, it is used for examining the contents of speech data files. For this function, the `TARGETKIND` configuration variable should not be set since no conversions of the data are required. Secondly, it is used for checking that input conversions are being performed properly. In the latter case, a configuration designed for a recognition system can be used with HLIST to make sure that the translation from the source data into the required observation structure is exactly as intended. To assist this, options are provided to split the input data into separate data streams (`-n`) and to explicitly list the identity of each parameter in an observation (`-o`).

6.5.2 Use

HList is invoked by typing the command line

```
HList [options] file ...
```

This causes the contents of each `file` to be listed to the standard output. If no files are given and the source format is `HAUDIO`, then the audio source is listed. The source form of the data can be converted and listed in a variety of target forms by appropriate settings of the configuration variables, in particular `TARGETKIND`¹.

The allowable options to HLIST are

- `-d` Force each observation to be listed as discrete VQ symbols. For this to be possible the source must be either `DISCRETE` or have an associated VQ table specified via the `VQTABLE` configuration variable.
- `-e N` End listing samples at sample index `N`.
- `-h` Print the source header information.
- `-i N` Print `N` items on each line.
- `-n N` Display the data split into `N` independent data streams.
- `-o` Show the observation structure. This identifies the rôle of each item in each sample vector.
- `-p` Playback the audio. When sourcing from an audio device, this option enables the playback buffer so that after displaying the sampled data, the captured audio is replayed.
- `-r` Print the raw data only. This is useful for exporting a file into a program which can only accept simple character format data.

¹The `TARGETKIND` is equivalent to the `HCOERCE` environment variable used in earlier versions of HTK

- `-s N` Start listing samples from sample index `N`. The first sample index is 0.
- `-t` Print the target header information.
- `-F fmt` Set the source data format to `fmt`.

HLIST also supports the standard options `-A`, `-C`, `-D`, `-S`, `-T`, and `-V` as described in section 8.4.

6.5.3 Tracing

HLIST supports the following trace options where each trace flag is given using an octal base

- 00001 basic progress reporting.

Trace flags are set using the `-T` option or the `TRACE` configuration variable.

6.6 HResults

6.6.1 Function

HRESULTS is the HTK performance analysis tool. It reads in a set of label files (typically output from a recognition tool such as HVITE) and compares them with the corresponding reference transcription files. For the analysis of speech recognition output, the comparison is based on a Dynamic Programming (DP) string alignment procedure. For the analysis of word-spotting output, the comparison uses the standard US NIST FOM metric.

When used to calculate the sentence accuracy using DP the basic output is recognition statistics for the whole file set in the format

```
----- Overall Results -----
SENT: %Correct=13.00 [H=13, S=87, N=100]
WORD: %Corr=53.36, Acc=44.90 [H=460,D=49,S=353,I=73,N=862]
=====
```

The first line gives the sentence-level accuracy based on the total number of label files which are identical to the transcription files. The second line is the word accuracy based on the DP matches between the label files and the transcriptions². In this second line, H is the number of correct labels, D is the number of deletions, S is the number of substitutions, I is the number of insertions and N is the total number of labels in the defining transcription files. The percentage number of labels correctly recognised is given by

$$\%Correct = \frac{H}{N} \times 100\% \quad (6.4)$$

and the accuracy is computed by

$$Accuracy = \frac{H - I}{N} \times 100\% \quad (6.5)$$

In addition to the standard HTK output format, HRESULTS provides an alternative similar to that used in the US NIST scoring package, i.e.

# Snt	Corr	Sub	Del	Ins	Err	S. Err
Sum/Avg	87	53.36	40.95	5.68	8.47	55.10 87.00

Optional extra outputs available from HRESULTS are

- recognition statistics on a per file basis
- recognition statistics on a per speaker basis
- recognition statistics from best of N alternatives

²The choice of “Sentence” and “Word” here is the usual case but is otherwise arbitrary. HRESULTS just compares label sequences. The sequences could be paragraphs, sentences, phrases or words, and the labels could be phrases, words, syllables or phones, etc. Options exist to change the output designations ‘SENT’ and ‘WORD’ to whatever is appropriate.

- time-aligned transcriptions
- confusion matrices

For comparison purposes, it is also possible to assign two labels to the same equivalence class (see `-e` option). Also, the *null* label `???` is defined so that making any label equivalent to the null label means that it will be ignored in the matching process. Note that the order of equivalence labels is important, to ensure that label `X` is ignored, the command line option `-e ??? X` would be used. Label files containing triphone labels of the form `A-B+C` can be optionally stripped down to just the class name `B` via the `-s` switch.

The word spotting mode of scoring can be used to calculate hits, false alarms and the associated figure of merit for each of a set of keywords. Optionally it can also calculate ROC information over a range of false alarm rates. A typical output is as follows

----- Figures of Merit -----				
KeyWord:	#Hits	#FAs	#Actual	FOM
A:	8	1	14	30.54
B:	4	2	14	15.27
Overall:	12	3	28	22.91

which shows the number of hits and false alarms (FA) for two keywords A and B. A label in the test file with start time t_s and end time t_e constitutes a hit if there is a corresponding label in the reference file such that $t_s < t_m < t_e$ where t_m is the mid-point of the reference label.

Note that for keyword scoring, the test transcriptions must include a score with each labelled word spot and all transcriptions must include boundary time information.

The FOM gives the % of hits averaged over the range 1 to 10 FA's per hour. This is calculated by first ordering all spots for a particular keyword according to the match score. Then for each FA rate f , the number of hits are counted starting from the top of the ordered list and stopping when f have been encountered. This corresponds to *a posteriori* setting of the keyword detection threshold and effectively gives an upper bound on keyword spotting performance.

6.6.2 Use

HRESULTS is invoked by typing the command line

```
HResults [options] hmmList recFiles ...
```

This causes HRESULTS to be applied to each `recFile` in turn. The `hmmList` should contain a list of all model names for which result information is required. Note, however, that since the context dependent parts of a label can be stripped, this list is not necessarily the same as the one used to perform the actual recognition. For each `recFile`, a transcription file with the same name but the extension `.lab` (or some user specified extension - see the `-X` option) is read in and matched with it. The `recfiles` may be master label files (MLFs)³. The available options are

³Reference MLFs should be supplied using the `-I` option.

- a s** change the label SENT in the output to **s**.
- b s** change the label WORD in the output to **s**.
- c** when comparing labels convert to upper case. Note that case is still significant for equivalences (see **-e** below).
- d N** search the first **N** alternatives for each test label file to find the most accurate match with the reference labels. Output results will be based on the most accurate match to allow NBest error rates to be found.
- e s t** the label **t** is made equivalent to the label **s**. More precisely, **t** is assigned to an equivalence class of which **s** is the identifying member.
- f** Normally, HRESULTS accumulates statistics for all input files and just outputs a summary on completion. This option forces match statistics to be output for each input test file.
- g fmt** This sets the test label format to **fmt**. If this is not set, the **recFiles** should be in the same format as the reference files.
- h** Output the results in the same format as US NIST scoring software.
- k s** Collect and output results on a speaker by speaker basis (as well as globally). **s** defines a pattern which is used to extract the speaker identifier from the test label file name. In addition to the pattern matching metacharacters ***** and **?** (which match zero or more characters and a single character respectively), the character **%** matches any character whilst including it as part of the speaker identifier.
- m N** Terminate after collecting statistics from the first **N** files.
- n** Set US NIST scoring software compatibility.
- p** This option causes a phoneme confusion matrix to be output.
- s** This option causes all phoneme labels with the form **A-B+C** to be converted to **B**. It is useful for analysing the results of phone recognisers using context dependent models.
- t** This option causes a time-aligned transcription of each test file to be output provided that it differs from the reference transcription file.
- u f** Changes the time unit for calculating false alarm rates (for word spotting scoring) to **f** hours (default is 1.0).
- w** Perform word spotting analysis rather than string accuracy calculation.

- `-z s` This redefines the null class name to `s`. The default null class name is `???`, which may be difficult to manage in shell script programming.
- `-G fmt` Set the label file format to `fmt`.
- `-I mlf` This loads the master label file `mlf`. This option may be repeated to load several MLFs.
- `-L dir` Search directory `dir` for label files (default is to search current directory).
- `-X ext` Set label file extension to `ext` (default is `lab`).

HRESULTS also supports the standard options `-A`, `-C`, `-D`, `-S`, `-T`, and `-V` as described in section 8.4.

6.6.3 Tracing

HRESULTS supports the following trace options where each trace flag is given using an octal base

- 00001 basic progress reporting.
- 00002 show error rate for each test alternative.
- 00004 show speaker identifier matches.
- 00010 warn about non-keywords found during word spotting.
- 00020 show detailed word spotting scores.
- 00040 show memory usage.

Trace flags are set using the `-T` option or the `TRACE` configuration variable.

6.7 HVite

6.7.1 Function

HVITE is a general-purpose Viterbi word recogniser. It will match a speech file against a network of HMMs and output a transcription for each. When performing N-best recognition a word level lattice containing multiple hypotheses can also be produced.

Either a word level lattice or a label file is read in and then expanded using the supplied dictionary to create a model based network. This allows arbitrary finite state word networks and simple forced alignment to be specified.

This expansion can be used to create context independent, word internal context dependent and cross word context dependent networks. The way in which the expansion is performed is determined automatically from the dictionary and HMMList. When all labels appearing in the dictionary are defined in the HMMList no expansion of model names is performed. Otherwise if all the labels in the dictionary can be satisfied by models dependent only upon word internal context these will be used else cross word context expansion will be performed. These defaults can be overridden by HNET configuration parameters.

HVITE supports shared parameters and appropriately pre-computes output probabilities. For increased processing speed, HVITE can optionally perform a beam search controlled by a user specified threshold (see `-t` option). When fully tied mixture models are used, observation pruning is also provided (see the `-c` option).

6.7.2 Use

HVITE is invoked via the command line

```
HVite [options] dictFile hmmList testFiles ...
```

HVite will then either load a single network file and match this against each of the test files `-w netFile`, or create a new network for each test file either from the corresponding label file `-a` or from a word lattice `-w`. When a new network is created for each test file the path name of the label (or lattice) file to load is determined from the test file name and the `-L` and `-X` options described below.

If no `testFiles` are specified the `-w s` option must be specified and recognition will be performed from direct audio.

The `hmmList` should contain a list of the models required to construct the network from the word level representation.

The recogniser output is written in the form of a label file whose path name is determined from the test file name and the `-l` and `-x` options described below. The list of test files can be stored in a script file if required.

When performing N-best recognition (see `-n N` option described below) the output label file can contain multiple alternatives `-n N M` and a lattice file containing multiple hypotheses can be produced.

The detailed operation of HVITE is controlled by the following command line options

- `-a` Perform alignment. HVITE will load a label file and create an alignment network for each test file.
- `-b s` Use `s` as the sentence boundary during alignment.

- c *f* Set the tied-mixture observation pruning threshold to *f*. When all mixtures of all models are tied to create a full tied-mixture system, the calculation of output probabilities is treated as a special case. Only those mixture component probabilities which fall within *f* of the maximum mixture component probability are used in calculating the state output probabilities (default 10.0).
- d *dir* This specifies the directory to search for the HMM definition files corresponding to the labels used in the recognition network.
- e When using direct audio input, output transcriptions are not normally saved. When this option is set, each output transcription is written to a file called *PnS* where *n* is an integer which increments with each output file, *P* and *S* are strings which are by default empty but can be set using the configuration variables *RECOUTPREFIX* and *RECOUTSUFFIX*.
- f During recognition keep track of full state alignment.
- g When using direct audio input, this option enables audio replay of each input utterance after it has been recognised.
- h *f* Load the TMF *f*. After the MMF has been loaded the MMF is transformed using the transformation set contained in the TMF, before recognition begins.
- i *s* Output transcriptions to MLF *s*.
- l *dir* This specifies the directory to store the output label files. If this option is not used then *HVITE* will store the label files in the same directory as the data. When output is directed to an MLF, this option can be used to add a path to each output file name. In particular, setting the option -l '*' will cause a label file named *xxx* to be prefixed by the pattern **/xxx* in the output MLF file. This is useful for generating MLFs which are independent of the location of the corresponding data files.
- m During recognition keep track of model boundaries.
- n *i* [*N*] Use *i* tokens in each state to perform N-best recognition. The number of alternative output hypotheses *N* defaults to 1.
- o *s* Choose how the output labels should be formatted. *s* is a string with certain letters (from *NSCTWMX*) indicating binary flags that control formatting options. *N* normalise acoustic scores by dividing by the duration (in frames) of the segment. *S* remove scores from output label. By default scores will be set to the total likelihood of the segment. *C* Set the transcription labels to start and end on frame centres. By default start times are set to the start time of the frame and end times are set to the end time of the frame. *T* Do not include times in output label files. *W* Do not include words in output label files when performing state or model alignment. *X* Strip triphones down to monophones. *M* Do not include model

names in output label files when performing state and model alignment.

- p f** Set the word insertion log probability to **f** (default 0.0).
- q s** Choose how the output lattice should be formatted. **s** is a string with certain letters (from **ABtvaldmm**) indicating binary flags that control formatting options. **A** attach word labels to arcs rather than nodes. **B** output lattices in binary for speed. **t** output node times. **v** output pronunciation information. **a** output acoustic likelihoods. **l** output language model likelihoods. **d** output word alignments (if available). **m** output within word alignment durations. **n** output within word alignment likelihoods.
- r f** Set the dictionary pronunciation probability scale factor to **f**. (default value 1.0).
- s f** Set the grammar scale factor to **f**. This factor post-multiplies the language model likelihoods from the word lattices. (default value 1.0).
- t f** Enable beam searching such that any model whose maximum log probability token falls more than **f** below the maximum for all models is deactivated. Setting **f** to 0.0 disables the beam search mechanism (default value 0.0).
- u i** Set the maximum number of active models to **i**. Setting **i** to 0 disables this limit (default 0).
- v f** Enable word end pruning. Do not propagate tokens from word end nodes that fall more than **f** below the maximum word end likelihood. (default 0.0).
- w [s]** Perform recognition from word level networks. If **s** is included then use it to define the network used for every file.
- x ext** This sets the extension to use for HMM definition files to **ext**.
- y ext** This sets the extension for output label files to **ext** (default **rec**).
- z ext** Enable output of lattices (if performing NBest recognition) with extension **ext** (default off).
- J tmf** Load the transform model file **tmf**. The transform is then applied to the acoustic model set before the recognition process begins.
- L dir** This specifies the directory to find input label (when **-a** is specified) or network files (when **-w** is specified).
- X s** Set the extension for the input label or network files to be **s** (default value **lab**).
- F fmt** Set the source data format to **fmt**.
- G fmt** Set the label file format to **fmt**.

- H *mmf* Load HMM macro model file *mmf*. This option may be repeated to load multiple MMFs.
- I *mlf* This loads the master label file *mlf*. This option may be repeated to load several MLFs.
- P *fmt* Set the target label format to *fmt*.

HVITE also supports the standard options -A, -C, -D, -S, -T, and -V as described in section 8.4.

6.7.3 Tracing

HVITE supports the following trace options where each trace flag is given using an octal base

- 0001 enable basic progress reporting.
- 0002 list observations.
- 0004 frame-by-frame best token.
- 0010 show memory usage at start and finish.
- 0020 show memory usage after each utterance.

Trace flags are set using the -T option or the TRACE configuration variable.

Part III

Language Modelling

Chapter 7

Introduction

The LVX decoder in the TRANSCRIBER package uses statistical language models as a secondary source of information when recognising speech utterances. The TRANSCRIBER distribution contains a number of tools for adapting, editing and testing statistical N-gram language models. The collection of tools will be referred to as the HTK Language Modelling Toolkit or HLM.

7.1 Introduction

An N-gram is a sequence of N symbols (e.g. words, syntactic categories, etc) and an N-gram language model (LM) is used to predict each symbol in the sequence given its $N - 1$ predecessors. It is built on the assumption that the probability of a specific N-gram occurring in some unknown test text can be estimated from the frequency of its occurrence in a training text. The *goodness* of a language model can be estimated by using it to compute a measure called *perplexity* on a previously unseen test set. In general, a good language model will have a low test-set perplexity.

Although the basic principle of an N-gram LM is very simple, in practice, there are usually many more potential N-grams than can ever be collected in a training text in sufficient numbers to yield robust frequency estimates. Furthermore, for any real application such as speech recognition, the use of an essentially static and finite training text makes it difficult to generate a single LM which is well-matched to varying test material. For example, an LM trained on newspaper text will be a good predictor for dictating business news reports but the same LM would be a poor predictor for personal letters or a spoken interface to a flight reservation system. A final difficulty is that the *vocabulary* of an N-gram LM is finite and fixed at construction time. Thus, if the LM is word-based, it can only predict words within its vocabulary and furthermore, new words cannot be added without rebuilding the complete LM.

7.2 N-Gram language models

The principle task of a statistical language model is to provide an estimate for the probability of a word sequence $W = w_1, w_2, \dots, w_K$. If it is assumed that the probability of each new word only depends on previous history, this probability can be

written as a product of conditional probabilities thus

$$P(W) = \prod_{k=1}^K p(w_k, |w_1, \dots, w_{k-1})$$

If the history used to determine each condition is now limited to just the preceding $N - 1$ words, an N-gram model results, ie.

$$P(W) = \prod_{k=1}^K p(w_k, |w_{k-N+1}, \dots, w_{k-1})$$

For speech recognition applications, the most common values for N are 2 (bigrams) and 3 (trigrams). The conditional N-gram probabilities are usually estimated from the relative frequency of occurrence of the corresponding events in the training text.

7.3 Perplexity

The quality of the language model can be evaluated by running a complete recognition experiment. Such evaluation has many advantages, the most important of which is the ability to observe the interaction between the acoustic model and the language model and to assess its effect on the overall search effort in the recogniser. Since this can be an expensive operation, one may consider the likelihood of the test data as given by the language model on its own. The test data perplexity is the likelihood if the test data renormalised with respect to the number of words N and is defined as

$$PLX = [p(w_1, w_2, \dots, w_N)]^{-1/N}$$

Using conditional probability and taking the log

$$\log PLX = -\frac{1}{N} \sum_{n=1}^N \log p(w_n | w_1, w_2, \dots, w_{n-1})$$

From the above, minimising the corpus perplexity is equivalent to maximising the log-likelihood of the test data. The definition of corpus perplexity introduces the coverage problem e.g. if a word in the vocabulary is assigned the probability of zero the perplexity measure will become infinitely large. Nevertheless, the perplexity measure of an unseen test set representative of the recognition task is used extensively to judge the quality of a language model.

7.4 LM tools in TRANSCRIBER

The TRANSCRIBER package provides three command-line tools for pruning, adapting and testing language models. The tool HLMCOPY enables the user to customise the supplied master models to a particular recognition task. The customisation process consists of pruning the supplied model to a specific vocabulary and optionally reducing its size in terms of the number of N-grams present in the target model. Furthermore, if text data representative of the target domain is available, the pruned model can be further adapted to that domain using the tool LADAPT. Finally, LPLEX can be used to evaluate the *goodness* of the generated models by computing the perplexity of unseen domain-specific text data.

Chapter 8

The Operating Environment

This chapter discusses the various ways of controlling the operation of the language modelling tools along with related aspects of file system organisation and error reporting.

The behaviour of a tool depends on three sources of information. Firstly, all command line tools are executed by issuing commands to the operating system shell. Each command typically contains the names of the various files that the tool needs to function and a number of optional arguments which control the detailed behaviour of the tool. Secondly, most command line tools use a set of standard library modules to interface to the various file types and to connect with the outside world. Many of these modules can be customised by setting parameters in a *configuration file*. Thirdly, a small number of parameters are specified using environment variables.

Terminal output mostly depends on the specific tool being used, however, there are some generic output functions which are provided by the library modules and which are therefore common across tools. These include version reporting, memory usage and error reporting.

Finally, the command line tools can read and write most data sources through pipes as an alternative to direct input and output from data files. This allows filters to be used, and in particular, it allows many of the external files to be stored directly in compressed form and then decompressed *on-the-fly* when the data is read back in.

All of the above is discussed in more detail in the following sections.

8.1 The Command Line

The general form of command line for invoking a tool is¹

```
tool [options] files ...
```

Options always consist of a dash followed by a single letter. Some options are followed by an argument as follows

-i	- a switch option
-t 3	- an integer valued option
-a 0.01	- a float valued option
-s hello	- a string valued option

¹All of the examples in this book assume the UNIX Operating System and the C Shell but the principles apply to any OS which supports hierarchical files and command line arguments

Option names consisting of a capital letter are common across all tools (see section 8.4). Integer arguments may be given in any of the standard C formats, for example, 13, 0xD and 015 all represent the same number. Typing the name of a tool on its own always causes a short summary of the command line options to be printed in place of its normal operation. For example, typing

```
HLMCopy
```

would result in the following output

```
USAGE: HLMCopy [options] inModel outModel
```

Option		Default
-d s	read source dict from s	none
-f s	set output LM format to s	bin
-c n c	set pruning for n-gram to c	0
-n n	save model as n-gram	max
-m	allow multiple identical prons	all
-o	copy first dict pron to output	all
-u s	read new unigrams from s	none
-v s	write pruned dict to s	none
-w fn	prune model using word list in fn	off
-A	Print command line arguments	Off
-C cf	Set config file to cf	default
-D	Display configuration variables	Off
-S f	Set script file to f	none
-T N	Set trace flags to N	0
-V	Print version information	Off

The first line shows the names of the required files and the rest consists of a listing of each option, its meaning, and its default value.

The precise naming convention for specifying files depends on the operating system being used, but the tools always assumes the existence of a hierarchical file system and it maintains a distinction between directory paths and file names.

In general, a file will be located either in the current directory, some subdirectory of the current directory or some subdirectory of the root directory. For example, in the command

```
LPlex -t lm1 dir/text1 /users/vv1/lm/text2
```

file `lm1` must be in the current directory, `text1` must be in the directory `dir` within the current directory and `text2` must be in the directory `/users/vv1/lm`.

8.2 Script Files

Tools which require a potentially very long list of files always allow the files to be specified in a script file via the `-S` option instead of via the command line. This is particularly useful when running under an OS with limited file name expansion capability. Thus, for example, LADAPT may be invoked by either

```
LAdapt -i 0.5 srcLM tgtLM s1 s2 s3 s4 s5 ....
```


or

```
LAdapt -S filelist -i 0.5 srcLM tgtLM
```

where `filelist` holds the list of files `s1`, `s2`, etc. Each file listed in a script should be separated by white space or a new line. Script files should only be used for storing ellipsed file list arguments. Note that shell meta-characters should not be used in script files and will not be interpreted by the tools.

8.3 Configuration Files

Configuration files are used for customising the HLM working environment. They consist of a list of parameter-values pairs along with an optional prefix which limits the scope of the parameter to a specific module or tool.

The name of a configuration file can be specified explicitly on the command line using the `-C` command. For example, when executing

```
LAdapt ... -C myconfig s1 s2 s3 s4 ...
```

The operation of `LADAPT` will depend on the parameter settings in the file `myconfig`.

When an explicit configuration file is specified, only those parameters mentioned in that file are actually changed and all other parameters retain their default values. These defaults are built-in. However, user-defined defaults can be set by assigning the name of a default configuration file to the environment variable `HCONFIG`. Thus, for example, using the UNIX C Shell, writing

```
setenv HCONFIG myconfig
LAdapt ... s1 s2 s3 s4 ...
```

would have an identical effect to the preceding example. However, in this case, a further refinement of the configuration values is possible since the opportunity to specify an explicit configuration file on the command line remains. For example, in

```
setenv HCONFIG myconfig
LAdapt ... -C xconfig s1 s2 s3 s4 ...
```

the parameter values in `xconfig` will override those in `myconfig` which in turn will override the built-in defaults. In practice, most users will set general-purpose default configuration values using `HCONFIG` and will then override these as required for specific tasks using the `-C` command line option.

The configuration file itself consists of a sequence of parameter definitions of the form

```
[MODULE:] PARAMETER = VALUE
```

One parameter definition is written per line and square brackets indicate that the module name is optional. Parameter definitions are not case sensitive but by convention they are written in upper case. A `#` character indicates that the rest of the line is a comment.

If the name of a configuration variable is mis-typed, there will be no warning and the variable will simply be ignored. To help guard against this, the standard option `-D` can be used. This displays all of the configuration variables before and after the tool runs. In the latter case, all configuration variables which are still unread are marked by a hash character.

8.4 Standard Options

Options consisting of a capital letter are common across all tools. The option `-C` is used to specify a configuration file name and the option `-S` is used to specify a script file name.

The three remaining standard options are `-A`, `-D` and `-V`. The option `-A` causes the current command line arguments to be printed. When running experiments via scripts, it is a good idea to use this option to record in a log file the precise settings used for each tool. The `-D` option causes the current configuration parameters set by the user to be displayed both before and after the tool has executed. The initial display allows the configuration values to be checked before potentially wasting a large amount of cpu time through incorrectly set parameters. The final display shows which configuration variables were actually used during the execution of the tool.

Finally, the option `-V` causes version information for the tool and each module used by that tool to be listed. These should always be quoted when making bug reports.

Finally, all tools implement the trace option `-T`. Trace values are typically bit strings and the meaning of each bit is described in the reference section for each tool. Setting a trace option via the command line overrides any setting for that same trace option in a configuration file. This is a general rule, command line options always override defaults set in configuration files.

All of the standard options are listed in the final summary section of this chapter.

8.5 Error Reporting

The LM tools rely on a standard mechanism for reporting errors and warnings. A typical error message is as follows

```
LAdapt: ERROR [+16410]
ProcessText: unable to open text file foo.txt
```

This indicates that the tool LADAPT is reporting an error number +16410. All errors have positive error numbers and always result in the tool terminating. Warnings have negative error numbers and the tool does not terminate.

The last two digits of an error number define the class of error. The remaining two or three digits indicate the module or a tool in which the error is located. The second line of the error message names the actual routine in which the error occurred (here `ProcessText`) and the actual error message. All errors and warnings are listed in chapter 10 indexed by their number. This listing contains more details on each error or warning along with suggested causes.

Error messages are sent to the standard error stream but warnings are sent to the standard output stream. The reason for the latter is that most tools are run with progress tracing enabled. Sending warnings to the standard output stream ensures that they are properly interleaved with the trace of progress so that it is easy to determine the point at which the warning was issued. Sending warnings to standard error would lose this information.

The default behaviour of a tool when terminating due to an error is to exit normally returning the error number as exit status. If, however, the configuration variable `ABORTONERR` is set true, then the tool will core dump. This is a debugging facility which should not concern most users.

8.6 Words and Strings

The HLM toolkit uses a variety of special notations representing model names, labels, etc., in which certain characters have a reserved meaning. Because of this, HLM uses a number of escaping conventions to allow it to handle arbitrary strings and names containing reserved characters. In HLM, a string consists of a single white space delimited word or a quoted string. Either the single quote ' or the double quote " can be used to quote strings but the start and end quotes must be matched. The backslash \ character can also be used to introduce otherwise reserved characters. The character following a backslash is inserted into the string without special processing unless that character is a digit in the range 0 to 7. In that case, the three characters following the backslash are read and interpreted as an octal character code. When the three characters are not octal digits the result is not well defined.

Although these conventions are unnecessary in HLM, to maintain compatibility with HTK, the same conventions are used. However, a number of options are provided to allow a mix of escaped and unescaped text files to be handled. Word maps allow the type of escaping (HTK or none) to be defined in their headers. When a degenerate form of word map is used (i.e. a map with no header), the LWMAP configuration variable `INWMAPRAW` may be set to true to disable HTK escaping. By default, HLM tools output word lists and maps in HTK escaped form. However, this can be overridden by setting the configuration variable `OUTWMAPRAW` to true.

8.7 LM file formats

Language models can be stored on disk in two different file formats - *binary* and *ultra*. The binary format is an Entropic-proprietary file format which is optimal in terms of flexibility and memory usage. Thus the master language models supplied in the TRANSCRIBER distribution are stored in this file format. The *ultra* LM format is a further development of the binary LM format optimised for fast loading times and small memory footprint. At the same time, models stored in this format cannot be pruned further in terms of size and vocabulary. All language models used in the example systems supplied with TRANSCRIBER are stored using the ultra format.

Input/output of N-gram language model files is handled by the HLM module `LModel`. By default, the tools `HLMCOPY` and `LADAPT` will save models using the binary file format. The `-f ultra` option will instruct the tools to save models in the ultra file format. Furthermore, all tools and the HAPI libraries in the TRANSCRIBER distribution are compatible with the ARPA-MIT file format used by toolkits such as the CMU LM toolkit.

8.8 Summary

This section summarises the environment variables and configuration parameters recognised by the tools. It also provides a list of all the standard command line options used with HLM.

Table 8.1 lists all of the configuration parameters along with a brief description. A missing module name means that it is recognised by more than one module. Table 8.2 lists all of the environment parameters used by these modules. Finally, table 8.3 lists all of the standard options.

Module	Name	Description
HShell	ABORTONERR	core dump on error (for debugging)
HShell	HLANGMODFILTER	filter for language model file input
HShell	HDICTFILTER	filter for Dictionary file input
HShell	HLANGMODOFILTER	filter for language model file output
HShell	HDICTOFILTER	filter for Dictionary file output
HShell	MAXTRYOPEN	number of file open retries
HShell	NONUMESCAPES	prevent string output using \012 format
	NATURALREADORDER	enable natural read order for binary files
	NATURALWRITEORDER	enable natural write order for binary files
	TRACE	trace control (default=0)
	STARTWORD	set sentence start symbol (<s>)
	ENDWORD	set sentence end symbol (</s>)
	UNKNOWNNAME	set OOV class symbol (!UNK)
	RAWMITFORMAT	disable HTK escaping
LWMap	INWMAAPRAW	disable HTK escaping for input word lists and maps
LWMap	OUTWMAAPRAW	disable HTK escaping for output word lists and maps

Table. 8.1 Configuration Parameters used in Operating Environment

Env Variable	Meaning
HCONFIG	Name of default configuration file
HxxxFILTER	Input/Output filters as above

Table. 8.2 Environment Variables used in Operating Environment

Standard Option	Meaning
-A	Print command line arguments
-C cf	Configuration file is cf
-D	Display configuration variables
-S scp	Use command line script file scp
-T N	Set trace level to N
-V	Print version information

Table. 8.3 Summary of Standard Options

Chapter 9

Language modelling tools

9.1 HLMCopy

9.1.1 Function

The basic function of this tool is to copy language models. During this operation the target model can be optionally adjusted to a specific vocabulary, reduced in size by applying pruning parameters to the different n -gram components and written out in a different file format. Previously unseen words can be added to the language model with unigram entries supplied in a unigram probability file. At the same time, the tool can be used to extract word pronunciations from a number of source dictionaries and output a target dictionary for a specified word list. HLMCOPY is a key utility in the TRANSCRIBER package enabling the user to construct custom dictionaries and language models tailored to a particular recognition task.

9.1.2 Use

HLMCOPY is invoked by the command line

```
HLMCopy [options] inLMFile outLMFile
```

This copies the language model `inLMFile` to `outLMFile` optionally applying editing operations controlled by the following options.

- d **fn** Use dictionary **fn** as a source of pronunciations for the output dictionary. A set of dictionaries can be specified, in order of priority, with multiple -d options.
- f **s** Set the output language model format to **s**. Possible options include **bin** for Entropic *binary* format and **ultra** for Entropic *ultra* format.
- c **n c** Set the pruning threshold for n -grams to **c**. Pruning can be applied to the bigram ($n=2$) and trigram ($n=3$) components of a model. The pruning procedure will keep only n -grams which have been observed more than **c** times. Note that this option is only applicable to master language models.
- n **n** Save target model as n -gram.
- m Allow multiple identical pronunciations for a single word. Normally identical pronunciations are deleted. This option may be required when a single word/pronunciation has several different output symbols.
- o Allow pronunciations for a single word to be selected from multiple dictionaries. Normally the dictionaries are prioritised by the order they appear on the command line with only entries in the first dictionary containing a pronunciation for a particular word being copied to the output dictionary.
- u **fn** Use unigrams from file **fn** as replacements for the ones in the language model itself. Any words appearing in the output language model which have entries in the unigram file (which is formatted as "BASE-10.LIKE WORD") use the likelihood ($\log_{10}(\text{prob})$)

from the unigram file rather than from the language model. This allows simple language model adaptation as well as allowing unigram probabilities to be assigned words in the output vocabulary that do not appear in the input language model.

- v fn** Write a dictionary covering the output vocabulary to file **fn**. If any required words cannot be found in the set of input dictionaries an error will be generated.
- w fn** Read a word-list defining the output vocabulary from **fn**. This will be used to select the vocabulary for both the output language model and output dictionary.

HLMCOPY also supports the standard options **-A**, **-C**, **-D**, **-S**, **-T**, and **-V** as described in section 8.4.

9.1.3 Tracing

HLMCOPY supports the following trace options where each trace flag is given using an octal base

- 00001 basic progress reporting.

Trace flags are set using the **-T** option or the **TRACE** configuration variable.

9.2 LAdapt

9.2.1 Function

This program will adapt an existing language model from supplied text data. This is accomplished in two stages. First, the text data is scanned and a new language model is generated. In the second stage, an existing model is loaded and adapted (merged) with the newly created one according to the specified ratio. The target model can be optionally pruned to a specific vocabulary.

9.2.2 Use

LADAPT is invoked by the command line

```
LAdapt [options] -i weight inLMFile outLMFile [texttfile ...]
```

The text data is scanned and a new LM generated. The input language model is then loaded and the two models merged. The effect of the weight (0.0-1.0) is to control the overall contribution of each model during the merging process. The output to outLMFile is an n -gram model stored in the user-specified format.

The allowable options to LADAPT are as follows

- b *n* Set the n -gram buffer size to *n*. This controls the size of the buffer used to accumulate n -gram statistics whilst scanning the input text. Larger buffer size will result in more efficient operation of the tool with fewer sort operations required.
- d *s* Set the root n -gram data file name to *s*. By default, n -gram statistics from the text data will be accumulated and stored as **gram.0**, **gram.1**, ..., etc. Note that a larger buffer size will result in fewer files.
- c *n c* Set the pruning threshold for n -grams to *c*. Pruning can be applied to the bigram ($n=2$) and trigram ($n=3$) components of the newly generated model. The pruning procedure will keep only n -grams which have been observed more than *c* times. Note that this option is only applicable to master language models.
- f *s* Set the output language model format to *s*. Possible options include **bin** for Entropic *binary* format and **ultra** for Entropic *ultra* format.
- g Use existing n -gram data files. If this option is specified the tool will use the existing gram files rather than scanning the actual texts. This option is useful when adapting multiple language models from the same text data or when experimenting with different merging weights.
- i *w fn* Interpolate with model *fn* using weight *w*. Note that at least one model must be specified with this option.
- n *n* Produce n -gram language model.
- s *s* Store *s* in the header of the gram files.

`-w fn` Load word list from `fn`. The word list will be used to define the target model's vocabulary. If a word list is not specified, the target model's vocabulary will have all words from the source model(s) together with any new words encountered in the text data.

LADAPT also supports the standard options `-A`, `-C`, `-D`, `-S`, `-T`, and `-V` as described in section 8.4.

9.2.3 Trace Output

LADAPT supports the following trace options where each trace flag is given using an octal base

00001 basic progress reporting.

Trace flags are set using the `-T` option or the `TRACE` configuration variable.

9.3 LPlex

9.3.1 Function

This program computes the perplexity and out of vocabulary (OOV) statistics of text data using one or more language models. The perplexity is calculated on per-utterance basis. Each utterance in the text data should start with a sentence start symbol (<s>) and finish with a sentence end (</s>) symbol. The default values for the sentence markers can be changed via the config parameters **STARTWORD** and **ENDWORD** respectively. Text data can be supplied as an HTK Master Label File (MLF) or as plain text (**-t** option). Multiple perplexity tests can be performed on the same texts using separate N-gram components of the model(s). OOV words in the test data can be handled in two ways. By default the probability of n -grams containing words not in the lexicon is simply not calculated. This is useful for testing closed vocabulary models on texts containing OOVs. If the **-u** option is specified, n -grams giving the probability of an OOV word conditioned on its predecessors are discarded, however, the probability of words in the lexicon can be conditioned on context including OOV words. The latter mode of operation relies on the presence of the unknown class symbol (!UNK) in the language model (the default value can be changed via the config parameter **UNKNOWNNAME**). If multiple models are specified (**-i** option) the probability of an n -gram will be calculated as a sum of the weighted probabilities from each of the models.

9.3.2 Use

LPLEX is invoked by the command line

```
LPlex [options] langmodel labelFiles ...
```

The allowable options to LPLEX are as follows

- c n c** Set the pruning threshold for n -grams to c . Pruning can be applied to the bigram ($n=2$) and trigram ($n=3$) components of the model. The pruning procedure will keep only n -grams which have been observed more than c times. Note that this option is only applicable to the model generated from the text data.
- e s t** Label **t** is made equivalent to label **s**. More precisely **t** is assigned to an equivalence class of which **s** is the identifying member. The equivalence mappings are applied to the text and should be used to map symbols in the text to symbols in the language model's vocabulary.
- i w fn** Interpolate with model **fn** using weight **w**.
- n n** Perform a perplexity test using the n -gram component of the model. Multiple tests can be specified. By default the tool will use the maximum value of n available.
- o** Print a sorted list of unique OOV words encountered in the text and their occurrence counts.
- t** Text stream mode. If this option is set, the specified test files will be assumed to contain plain text.

- u** In this mode OOV words can be present in the n -gram context when predicting words in the vocabulary. The conditional probability of OOV words is still ignored.
- w fn** Load word list in **fn**. The word list will be used as the restricting vocabulary for the perplexity calculation. If a word list file is not specified, the target vocabulary will be constructed by combining the vocabularies of all specified language models.
- z s** Redefine the null equivalence class name to **s**. The default null class name is **???**. Any words mapped to the null class will be deleted from the text.

LADAPT also supports the standard options **-A**, **-C**, **-D**, **-S**, **-T**, and **-V** as described in section 8.4.

9.3.3 Trace Output

LADAPT supports the following trace options where each trace flag is given using an octal base

- 00001 basic progress reporting.

Trace flags are set using the **-T** option or the **TRACE** configuration variable.

Chapter 10

Error and Warning Codes

When a problem occurs in any HLM tool, either an error or warning message is printed. The format of these messages is as follows

```
Tool: WARNING [-nnnn]
      Function: Brief explanation
```

```
Tool: ERROR [+nnnn]
      Function: Brief explanation
```

The message consists of four parts. On the first line is the tool name and the error number. On the second line is the function in which the problem occurred and a brief textual explanation. When an error occurs the message is printed to standard error and the program terminates immediately. For warnings, the message is sent to standard output and execution continues. The reason for sending warnings to standard output is so that they are synchronised with any trace output.

Error numbers in HLM are allocated on a module by module and tool by tool basis in blocks of 100 as shown by the table shown overleaf. Within each block of 100 numbers the first 20 (0 - 19) and the final 10 (90-99) are reserved for standard types of error which are common to all tools and library modules. All other codes are module or tool specific.

10.1 Generic Errors

- + [?]??01 Facility not implemented
HLM does not support the operation requested.
- + [?]??05 Available memory exhausted
The operation requires more memory than is available.
- + [?]??10 Cannot open file for reading
Specified file could not be opened for reading. The file may not exist or the filter through which it is read may not be set correctly.
- + [?]??11 Cannot open file for writing
Specified file could not be opened for writing. The directory may not exist or be writable by the user or the filter through which the file is written may not be set correctly.

- + [?]??13 Cannot read from file
Cannot read data from file. The file may have been truncated, incorrectly formatted or the filter process may have died.
- + [?]??14 Cannot write to file
Cannot write data to file. The file system is full or the filter process has died.
- + [?]??15 Required function parameter not set
You have called a library routine without setting one of the arguments.
- + [?]??16 Memory heap of incorrect type
Some library routines require you to pass them a heap of a particular type.
- + [?]??19 Command line syntax error
The command line is badly formed, refer to the manual or the command summary printed when the command is executed without arguments.
- + [?]??9? Sanity check failed
Several functions perform checks that structures are self consistent and that everything is functioning correctly. When these sanity checks fail they indicate the code is not functioning as intended. These errors should not occur and are not correctable by the user.

LAdapt	16400-16499	LWMap	15100-15199
LPlex	16600-16699	LUtil	15200-15299
HLMCopy	16900-16499	LGBase	15300-15399
HShell	5000-5099	LModel	15400-15499
HMem	5100-5199	LPCalc	15500-15599
HMath	5200-5299	LPMerge	15600-15699
HLabel	6500-6599		
HDict	8000-8099		

10.2 Summary of Errors by Tool and Module

For information on errors reported by the modules HMEM, HSHELL, HLABEL and HDICT the reader should refer to the HAPI Book.

LPLEX

- +16620 symbol XYZ not in word list
The sentence start symbol, sentence end symbol and OOV symbol (only if OOVs are to be included in the perplexity calculation) must be in the language model's vocabulary. Note that the vocabulary list is either specified with the `-w` option or is implicitly derived from the language model.

- +16625 Unable to find word XYZ in any model
Ensure that all words in the vocabulary list specified with the `-w` option are present in at least one of the language models.
- +16630 Maximum number of unique OOV's reached
Too many OOVs encountered in the input text.
- 16635 Transcription file `fn` is empty
The label file does not contain any words.
- 16640 Word too long, will be split: XYZ
The word read from the input stream is of over 200 characters.
- 16645 Text buffer size exceeded (NNN)
The maximum number of words allowed in a single utterance has been reached.

HLMCOPY

- +16920 Maximum number of phones reached
When HLMCOPY is used to copy dictionaries, the target dictionary's phone table is composed by combining the phone tables of all source dictionaries. Check that the number of different phones resulting from combining the phone tables of the source dictionaries does not exceed the internally set limit.
- +16930 Cannot find definition for word XYZ
When copying dictionaries, ensure that each word in the vocabulary list occurs in at least one source dictionary.

LWMAP

- +15150 word list/word map file format error
Check that the word list/word map file is correctly formatted.

LUTIL

- +15250 Header format error
Ensure that word maps and/or n-gram files used by the program start with the appropriate header.

LGBASE

- +15330 n-gram file consistency check failure
The n-gram file is incompatible with other resources used by the program.
- +15350 n-gram file format error
Ensure that n-gram files used by the program are formatted correctly and start with the appropriate header.

LMODEL

- +15420 cannot find n-gram component
The internal structure of the language model is corrupted. This error is usually caused when an n-gram (a, b, c) is encountered without the presence of n-gram (a, b) .

- +15430 incompatible probability kind in conversion
The currently used language model does not allow the required conversion operation. This error is caused by attempting to prune a model stored in the ultra file format.
- +15440 cannot prune models in ultra format
Pruning of language models stored in *ultra* file format is not supported.
- +15445 word ID size error
Language models with vocabularies of over 65,536 words require the use of larger word identifiers. This is a sanity check error.
- +15450 language model file format error
The language model file is formatted incorrectly. Check the file is complete and has not been corrupted.
- −15460 model order reduced
Due to the effects of pruning the model order is automatically reduced.

LPCALC

- +15520 unable to find FLEntry to attach
Indicates that the LM data structures are corrupt. This is normally caused by NGram files which have not been sorted.
- +15525 attempt to overwrite entries when attaching
Indicates that the LM structure is corrupt. Ensure that the word map file used is suitable for decoding the NGram database files.
- +15540 pruning error
The pruning parameters specified are not compatible with the parameters of the language model.

LPMERGE

- +15620 unable to find word in any model
Indicates that the target model vocabulary contains a word which cannot be found in any of the source models.

Part IV

Appendices

Appendix A

Acoustic models

The TRANSCRIBER distribution contains a number of acoustic model sets for each of the supported language variants. These have been trained using several hours of speech data collected from a large number of speakers, recorded using close-talking microphones in low-noise environments. The model sets will provide best performance if the conditions during recognition match the training conditions as closely as possible.

The acoustic model sets model the speech signal at subword unit (phone) level. All sets contain cross-word context dependent (triphone) models suitable for use with the large vocabulary (LVX) decoder in the HAPI library. Up to three model sets are supplied for each of the supported languages. These differ in terms of their size and allow the developer to achieve an optimal balance between performance and memory footprint. Table A.1 lists the available clean-speech model sets for the supported languages.

Language	Small	Medium	Large
UK English	EPCX1_UK01.mmf	EPCX2_UK01.mmf	n/a
US English	EPCX1_US01.mmf	EPCX2_US01.mmf	EPCX3_US01.mmf
UX English	EPCX1_UX01.mmf	EPCX2_UX01.mmf	EPCX3_UX01.mmf

Table A.1: UK, US and UX acoustic model sets

The small sets are intended as seed models for speaker dependent model sets derived using the Enrollment tool in TRANSCRIBER and/or the command line tool HAPIAdapt. Medium-sized model sets are typically used in speaker independent systems. However, as the amount of adaptation data grows these model sets become equally well suited to speaker dependent/adapted system. Large model sets should be used in systems where best accuracy is required (with some increase in memory footprint and time used to perform recognition).

Note that all clean-speech model sets require 16KHz-sampled waveform data. The HAPI configuration parameters required to setup the front-end coder for use with these models reside in `config/coding16epc.cfg`.

A.1 Telephony model sets

The TRANSCRIBER CDROM distribution also includes acoustic model sets suitable for speech recognition over the telephone. These model sets are not part of the default installation. If required, the user should install them separately from the “telpack” add-on.

Language	Small	Medium	Large
UK English	n/a	n/a	n/a
US English	n/a	EPCX2_UST1.mmf	n/a
UX English	n/a	n/a	n/a

Table A.2: UK, US and UX acoustic model sets

Note that all telephony model sets have been designed for use with waveform data sampled at 8KHz. The HAPI configuration parameters required to setup the front-end coder for use with these models reside in `config/coding8epc.cfg`.

Appendix B

Language models

The TRANSCRIBER distribution includes a trigram language model covering over 200,000 of the most common words and names in English. The model was derived from approximately 0.5 billion words of text in the North American Business news domain. The language model (`ECRL_BN1_200K.lm`) resides in the `resources` directory of the TRANSCRIBER distribution. The model cannot be used directly in a speech recognition system and forms a generic (“master”) resource from which task-specific language models can be derived using the command line tool `HLMCopy`. In addition, the following files are supplied which aim to help the developer in customising the “master” model to a specific task domain.

- `ECRL_200K.wlist`
A list of all words which appear in the 200K language model and dictionaries.
- `ECRL_200K.freq`
Word frequency list for the 200K word list. Counts are occurrences per million words.
- `ECRL_200K.uni`
Unigram $\log_{10}()$ likelihoods for the 200K word list. These are provided to help estimate values for previously unseen words.
- `ECRL_200K.pinfo`
This file tabulates the possible pruning parameter values and the number of bigram and trigram entries that will remain in the language model after pruning using `HLMCopy`.

B.1 Mixed-case language model

The TRANSCRIBER distribution includes a 200,000 word mixed-case language model for use with the mixed-case dictionaries described in appendix C. This model is not part of the default installation. If required, the user should install it separately from the “lmpack” add-on.

The mixed case model (`ECRL_BN1_200KC.lm`) was derived from approximately 1 billion words of text in the North American Business/general news domain. The following additional resources for the 200KC model are also available:

- ECRL_200KC.wlist
A list of all words which appear in the 200KC language model and dictionaries.
- ECRL_200KC.freq
Word frequency list for the 200KC word list. Counts are occurrences per million words.
- ECRL_200KC.uni
Unigram $\log_{10}()$ likelihoods for the 200KC word list. These are provided to help estimate values for previously unseen words.
- ECRL_200KC.pinfo
This file tabulates the possible pruning parameter values and the number of bigram and trigram entries that will remain in the language model after pruning using HLMCopy.

Appendix C

Dictionaries

This appendix provides details of the UK, US and UX TRANSCRIBER dictionaries. The dictionaries cover approximately 200,000 of the most common words and names in English. The dictionaries reside in the `resources` directory of the TRANSCRIBER distribution:

```
resources/ECRL_UK_200K.dct
resources/ECRL_US_200K.dct
resources/ECRL_UX_200K.dct
```

Table C.1 shows the number of distinct words and pronunciations in the UK, US and UX English dictionaries.

Word Breakdown	UK Dict	US Dict	UX Dict
Unique Words	203,866	203,866	203,866
Pronunciations	235,371	234,284	235,371

Table C.1: UK, US and UX dictionary breakdown

C.1 Output Symbols

In all supplied dictionaries, the default output symbol for most words is identical to the main word entry. The few exceptions include the sentence start and end symbols, `<s>` and `</s>`. These special symbols map to silence, and have no output symbol in the recognised word string. Furthermore, all punctuation words have the actual punctuation mark as an output symbol.

C.2 Phone Sets

Each dictionary entry has an associated pronunciation represented by a sequence of phones. Each phone represents a distinct sound in the language. In addition, two phones **sil** and **sp** can be used to represent silence. The first model (**sil**) is used for long pauses or to model silence at the start and end of sentences. The second model (**sp**) is used for optional pauses between words. Note that **sil** and **sp** do not appear

explicitly at the end of each pronunciation. The LVX decoder will automatically derive two pronunciations from each dictionary entry by adding the silence phones. Similar behaviour can be achieved when using the standard HTK decoder by setting the `ADDSILPHONES` config parameter to `HNet`.

The UK, US and UX dictionaries use a phone set of 41 speech phones and 2 silence phones. The US phone set is illustrated below

Symbol	Example	Symbol	Example
Vowels		Plosives	
aa	b <u>a</u> lm	b	<u>b</u> et
aa	b <u>o</u> x	d	<u>d</u> ebt
ae	b <u>a</u> t	g	<u>g</u> et
ah	b <u>u</u> t	k	<u>c</u> at
ao	b <u>o</u> ught	p	<u>p</u> et
aw	b <u>o</u> ut	t	<u>t</u> at
ax	<u>a</u> bout	Fricatives	
ay	b <u>i</u> te	dh	<u>t</u> hat
eh	b <u>e</u> t	th	<u>t</u> hin
er	b <u>i</u> rd	f	<u>f</u> an
ey	b <u>a</u> it	v	<u>v</u> an
ih	b <u>i</u> t	s	<u>s</u> ue
iy	b <u>e</u> et	sh	<u>sh</u> oe
ow	b <u>o</u> at	z	<u>z</u> oo
oy	b <u>o</u> y	zh	mea <u>s</u> ure
uh	b <u>o</u> ok	Affricates	
uw	b <u>o</u> ot	ch	<u>ch</u> ea <u>p</u>
Glides		jh	<u>j</u> ee <u>p</u>
l	<u>l</u> ed	Nasals	
r	<u>r</u> ed	m	<u>m</u> et
w	<u>w</u> ed	n	<u>n</u> et
y	<u>y</u> et	en	bu <u>t</u> ton
hh	<u>h</u> at	ng	th <u>ng</u>
		Silence	
		sil	silence
		sp	short pause

Table C.2: TRANSCRIBER US Phone Set

In contrast with the US phone set, the UK/UX dictionary does not use the phone **en**, this is expanded to the phone sequence **ax n**. The UK dictionary uses an additional phone **oh** as found in *hot*. This is distinct from the vowel **aa** in *heart*, whereas the US dictionary makes no distinction, and uses the same vowel **aa** in both cases.

C.3 Mixed-case dictionaries

The TRANSCRIBER distribution includes a set of mixed-case dictionaries for applications which require output of mixed case text. These dictionaries are not part of

Symbol	Example	Symbol	Example
Vowels		Plosives	
aa	ba <u>l</u> m	b	<u>b</u> et
aa	ba <u>r</u> n	d	<u>d</u> ebt
ae	ba <u>t</u>	g	<u>g</u> et
ah	bu <u>t</u>	k	<u>c</u> at
ao	bo <u>u</u> ght	p	<u>p</u> et
aw	bo <u>u</u> t	t	<u>t</u> at
ax	<u>a</u> bout	Fricatives	
ay	bi <u>t</u> e	dh	<u>th</u> at
eh	<u>b</u> et	th	<u>th</u> in
er	bi <u>r</u> d	f	<u>f</u> an
ey	ba <u>i</u> t	v	<u>v</u> an
ih	bi <u>t</u>	s	<u>s</u> ue
iy	be <u>e</u> t	sh	<u>sh</u> oe
oh	bo <u>x</u>	z	<u>z</u> oo
ow	bo <u>o</u> t	zh	mea <u>s</u> ure
oy	bo <u>y</u>	Affricates	
uh	bo <u>o</u> k	ch	<u>ch</u> ea <u>p</u>
uw	bo <u>o</u> t	jh	<u>j</u> ee <u>p</u>
Glides		Nasals	
l	<u>l</u> ed	m	<u>m</u> et
r	<u>r</u> ed	n	<u>n</u> et
w	<u>w</u> ed	ng	th <u>ng</u>
y	<u>y</u> et	Silence	
hh	<u>h</u> at	sil	silence
		sp	short pause

Table C.3: TRANSCRIBER UK/UX Phone Set

the default TRANSCRIBER installation and if required the user should install them separately.

```
resources/ECRL_UK_200KC.dct
resources/ECRL_US_200KC.dct
resources/ECRL_UX_200KC.dct
```

Table C.4 shows the number of distinct words and pronunciations in the UK, US and UX mixed case English dictionaries.

Word Breakdown	UK Dict	US Dict	UX Dict
Unique Words	209,926	209,926	209,926
Pronunciations	242,050	240,892	242,045

Table C.4: Mixed-case UK, US and UX dictionary breakdown

Index

- ABORTONERR, 84
- acoustic models, 101
 - clean speech, 101
 - telephony speech, 102
- adaptation, 15
 - evaluating, 17
 - incremental, 19
 - initialising models, 20
 - supervised, 15
 - types, 17
 - unsupervised, 16
- ARPA-MIT LM format, 85
- command line
 - arguments, 81
 - integer argument formats, 82
 - options, 81
- confidence score, 42
- configuration files, 83
 - default, 83
 - format, 83
- configuration parameters
 - operating environment, 85
- dictionary
 - mixed case, 106
 - output symbols, 105
 - system
 - UK English, **105**
 - US English, **105**
- Enrollment tool, 35
 - alignment, 42
 - auto increment, 42
 - collecting data, 40
 - confidence score, 42
 - controls, 39
 - menu bar, 37
 - resuming a session, 43
 - terminating a session, 42
 - waveform display, 39
- environment variables, 85
- error message
 - format, 95
- error number
 - structure of, 95
- error numbers
 - structure of, 84
- errors, 84
 - full listing, 95
- files
 - configuration, 83
 - language models, 85
 - script, 82
- HAPIADAPT, 46–47
- HAPIVITE, 48–51
- HCONFIG, 83
- HCOPY, 52–54
- HHED, 55–65
- HLIST, 66–67
- HRESULTS, 68–71
- HVITE, 72–75
- LADAPT, 90–91
- language models, 103
 - mixed case, 103
- LM file formats
 - ARPA-MIT format, 85
 - binary, 85
 - ultra, 85
- LPLEX, 92–93
- phone sets, 105
- pipes, 81
- prompt file, 41
- script files, 82
- standard options, 84
 - A, 84
 - C, 83, 84
 - D, 84

- S, 82, 84
 - T, 84
 - V, 84
 - summary, 86
- strings
 - metacharacters in, 85
- System Evaluation tool
 - enrollment, 34
- System evaluation tool, 23
 - calibration, 29
 - configuration files, 26
 - N-best output, 32
 - playback, 32
 - recogniser configuration, 26
 - source settings, 29
 - speaker adaptation, 33
 - word-output mode, 23, 33
- termination, 84
- tracing, 84
- warning codes
 - full listing, 95
- warning message
 - format, 95
- warnings, 84