TR-IT-0146

Software for Design and Use of a Speech Recognition System based on Automatically Derived Non-Uniform Units

Michiel Bacchiani Mari Ostendorf

1996.01

In this report, the software produced to implement the design and use of a speech recognition system as described in technical report TR-IT-0147 is described. It describes three software packages which implement automatic unit unit design, lexicon building and pronunciation modeling.

1 Introduction

This technical report describes the software that was produced to design and use a speech recognition system, based on automatically derived acoustic units. The software described in this report covers the available software at this time, January 1996. The software was developed between March 1995 and January 1996. The theoretical description of the tasks performed by the software described in this report can be found in technical report TR-IT-0147, "Speech Recognition System Design using Automatically Learned Non-Uniform Segmental Units" by M. Ostendorf, M. Bacchiani, Y. Sagisaka and K. Paliwal. This report will be refered to as "TR-IT-0147" from now on.

The software sets by default the parameters, which can be manipulated through command line options, to typical values.

2 Software description

The currently available software consists of 3 packages. The first package implements the unit design algorithm described in TR-IT-0147 and is named *ASSM*. The second package implements the lexicon building algorithm described in TR-IT-0147 and is named *LEXBLD*. The third package implements some of the ideas on pronunciation modeling described in TR-IT-0147 and is named *CMPM*.

To compile each of these packages, the following is required:

- A C compiler (all packages were only tested using gnu's C compiler `gcc` though).
- The HTK libraries.

Each of the packages are archived using the UNIX `tar` utility. The archives are compressed using the the GNU compression utility `gzip`. The files, neccessary for compiling the package can be extracted directly from the compressed archive using the GNU `tar` utility by the command:

```
tar xvfz <compressed package archive>
```

where `<compressed package archive>` denotes the name of the compressed archive that is to be unpacked. This process will build a directory structure containing the files of the package. In this directory structure, the source files for a particular software tool are found in a subdirectory with the same name as the software tool. In the top directory, a file can be found with the name `Make.glob`. Only this file needs to be edited before compilation to define where the different resources (i.e. libraries) can be found and where the executable should be placed. All the `Make.glob` for the 3 software packages have the lines shown in the following example configuration:

```
MACHINE  = LINUX
CC       = gcc
OPTLIB   = /usr/local/lib/HTKLib.P90.a
HTKHDRS  = /usr/local/include/HTK/
DESTDIR  = /home/chiel/src/LEXBLD_1.0/bin/
```

The `MACHINE` line can take the values `HP, SUN, DEC, LINUX` for the HP-UX, SUN4, DEC OSF1 and Linux operating systems respectively. The `CC` defines the C compiler that is to be used. The `OPTLIB` should define the name of the HTK library or libraries that should be used in linking. The `HTKHDRS` line specifies the location of the HTK header files. The `DESTDIR` defines where the resulting executables should be placed. For the *ASSM* package, there is an extra line in the `Make.glob` file which specifies where the files `HRmIO.h` and `HRmIO.c` can be found. This line is of the form `HRMIODIR = <directory>`. For the *CMPM* package, there is an extra line in the `Make.glob` file which specifies where the files `BLexIO.h`, `BLexIO.c`, `BfMdlIO.h` and `BfMdlIO.c` can be found. This line is of the form `IOMHDRS = <directory>`. After editing the `Make.glob` file, all software tools will be compiled by simply giving the command `make` in the top directory.

The different packages and the software tools found in these packages are described in detail in the following subsections.

3 Unit design: ASSM

The *ASSM* package implements the unit design algorithm described in TR-IT-0147. It consists of 4 core programs which perform the different steps of the unit design algorithm. Besides these 4 core programs, a number of utilities are included which allow the user to investigate the results produced by the 4 core programs. The following subsection gives a brief overview of the 4 core programs and the information and file formats that these programs use to interact with each other. Then, in the subsequent subsection, a more detailed description of the core programs and the utilities will follow.

3.1 Core programs and file formats

The first step of the algorithm performs the acoustic segmentation. The program that implements this is named *asegm*. This program takes an utterance in the form of a series of feature vectors as input and produces 2 types of files as output. As the HTK library is used to build the programs, the input file is assumed to be in the HTK file format. For details on this format, please look in the documentation that comes with the HTK package.

The 2 output file types are a label file which indicates where the segment boundaries are placed and a *SegFile* format file which contains sufficient statistic representations of the resulting segments. The default output format of the label file is HTK format but other file formats are supported. For a detailed description of the supported file formats for the label file, please look in the HTK documentation. Note that all the functionality for input and output files, found in the HTK tools are also available for the software described here (i.e. input and output file formats can be manipulated by the setting of environment variables).

The *SegFile* format file contains information on the segmented utterance. It contains:

- the number of segments found in the utterance
- the dimension of the feature vectors
- the regression order used in the segmentation
- the regression order of the sufficient statistics of the resulting segments
- the frame duration
- the time offset of the first frame in the utterance
- the distance type used in the segmentation
- the covariance type of the sufficient statistics of the resulting segments
- the segment durations in frames
- the segment mean trajectory parameters (**B** matrix)
- the segment covariances
- the file header size

The second step of the algorithm is to cluster the segments obtained from the acoustic segmentation. The program that implements this is named ***km-las***, and it uses a binary splitting routine to get an initial set of M models that are then refined by iterative re-estimation. This program takes *SegFile* format files as its input and produces unit model files as output. The unit models are described by a file format named *RegModelFile*. The *RegModelFile* contains the following information:

- the dimension of the feature vectors
- the regression order of the model
- the number of segments used to estimate the model parameters

- the number of frames in the segments used to estimate the model parameters
- the minimum and maximum duration in frames among the segments used to estimate the model parameters
- the covariance type of the model
- the relative frequency of the segment lengths among the segments used to estimate the model parameters
- the segment mean trajectory parameters (**B** matrix)
- the covariance of the model

The third step of the unit design algorithm performs a Viterbi re-segmentation of an utterance using a model inventory. The program that implements this is named *vitsegm*. It takes a collection of *RegModelFile* format files, an utterance represented by feature vectors and an initial segmentation file (in the label file format) as inputs and produces a new segmentation file (label file format) as output. The model inventory is represented by the collection of *RegModelFile* format files. The input label file of the utterance is used to constrain the search space of the re-segmentation process to reduce computational costs. The output label file describes the new segment boundaries and segment identities in terms of the model inventory. The re-segmentation can be constrained further by specifying a boundary constraint file. The boundary constraints dictate that a segment boundary in the resulting segmentation is found at the specified fixed time. This feature is incorporated to facilitate re-segmentation given boundary constraints imposed by the lexicon building algorithm. The format of such a file is equal to a label file. The "label boundary" times indicate the boundary constraints that are to be imposed. If an odd number of boundary constraints is to be imposed, the last "label" definition in this boundary constraint file will have equal boundary times. All functionality found in the HTK package with respect to input and output of label files also applies to this program.

The fourth step of the algorithm is implemented by a program that is named *reclust*. The inputs to the *reclust* program are a previous model inventory, represented by a collection of *RegModelFile* format files, and a collection of label files with the corresponding feature vector representation of these utterances. The output is a new model inventory, also represented by a collection of *RegModelFile* format files. The new model inventory is estimated from the segments specified by the label files. Again, all HTK file input and output label file functionality also applies to this program.

The *reclust* produces a bigram probability file in addition to the new model inventory. The bigram probability file is an optional to the *vitsegm* program. The bigram file is in text format and each line specifies one bigram probability in the log domain. The format of such a line is:

```
context target probability
```

This line specifies that $\log(P(target|context)) = probability$. As an example, the line

```
Unit3 Unit16 -4.3685519
```

indicates that the log-probability of Unit16 in the context of Unit3 is -4.3685519.

3.2 The acoustic segmentation program: asegm

Executing asegm with no command line arguments gives the following output:

```
USAGE: asegm [options] filenames...
```

```
Option                                          Default

-c s      Set covariance type of models to s FULLCOV
-d n f1 f2....fn
          Set n distortion threshold(s) to
          f1,...,fn                          1 1.0
-m i      Set minimum segmentlength          2 frames
-n i      Set maximum segmentlength          70 frames
-v        Verbose                            off
-w f      Parameterisation window width (ms) 25.6
-C s      Set regression model dir. to s     current
-D s      Set distortion measure to type s   MAHALANOBIS
-F fmt    Set data file format to fmt        HTK
-L s      Set segmentation info dir. to s    current
-R i      Set segmentation-regression order  0
-S f      Set script file to f               none
-W i      Set model regression order         0
-X s      Set extension for segm. info files alab.<dist>
-Z s      Set extension for model files      mdl.<dist>

Distance measures:
   SQ_EUCLID   for squared Euclidean distance
   MAHALANOBIS for Mahalanobis distance

Covariance types:
   DIAGONALCOV   for diagonal covariance
   FULLCOV       for full covariance
```

The files containing the feature vector representations of the utterances that are to be segmented have to be specified on the command line after the options.

The **S** switch allows the user to specify a text file which contains a list of filenames of files to be processed. The list of filenames is appended to the command line and is available to avoid problems with command lines extending over the maximum size allowed by the operating system.

The allowable average distance per frame controls the number of segments in the resulting segmentation. The value for this threshold is set using the **d** switch. This switch also allows one to write out the segmentation corresponding to a number of different thresholds. The first argument of the switch specifies the number of segmentations that are desired, the following arguments specify the threshold values for these segmentation. For example, if the segmentation for the thresholds 3, 4 and 2.37 are desired, the **d** switch part of the command line should look like `-d 3 3 4 2.37`.

The directory where the label files are written is controlled by the **L** switch, the directory where the *SegFile* format files are written is controlled by the **C** switch.

The extension of the label files are by default alab.*D* where *D* indicates the value of the distance threshold set by the **d** switch. For example, if a file **x.mfc** is segmented using thresholds 1 and 7, the default setting will generate 2 label files with names **x.alab.1** and **x.alab.7**. The extension of the label files can be manipulated using the **X** switch. The extensions of *SegFile* format files are derived in a similar way and can be manipulated through the **Z** switch.

3.3 The clustering program: km-las

Executing km-las with no command line arguments gives the following output:

```
USAGE: km-las [options] filenames...

 Option                                        Default

 -c s    Set the covariance to type s          FULLCOV
 -d s    Set distance measure to s             MAXLIKELIHOOD
 -e f    Set splitting parameter to f          0.01
 -f f    Set variance floor to f               1e-05
 -m i    Set minimum clustersize in frames     25
 -t f    Set convergence threshold to f        0.001
 -v      Verbose                               off
 -B s    Set codebooks directory to s          current
 -C i    Set codebook size to i                1000
 -D i    Set codebook export interval to i     10
 -I s    Set initial modellist to s            none
 -J s    Set initial models directory to s     current
 -M i    Set max. nr. of bin. split. iters.    10
 -N i    Set max. nr. of K-means iters.        30
 -S f    Set script file to f                  none
 -U s    Set new units name to s               Unit
 -V i    Set start index of new units to i     1

Distance measures:
   SQ_EUCLID     for squared Euclidean distance
   MAHALANOBIS   for Mahalanobis distance
   MAXLIKELIHOOD for MaximumLikelihood distance

Covariance types:
   DIAGONALCOV   for diagonal covariance
   FULLCOV       for full covariance
```

The *SegFile* format files containing the sufficient statistics representations of the segments that are to be clustered have to be specified on the command line after the options. The **S** allows the expansion of the command line as described under the description of the *asegm* program.

The desired number of models is specified by the **C** switch.

The final model inventory is written in a directory called "codebook.km" which will be made in the directory specified by the **B** switch. The model inventory during the binary split clustering can also be written to this directory every time the model inventory has increased with N models. The value for N can be specified by the **I** switch. The model inventories during the binary split clustering are written in a directory named "codebook.M" where M indicates the size of the model inventory.

The maximum number of iterations that are performed during each step of the binary split clustering is specified by the **M** switch. The maximum number of iterations that are performed during the final full K-means clustering is specified by the **N** switch. The iterations in both types of clustering steps are stopped as soon as the decrease in summed distance is less than a fraction of the last distance. This fraction is specified by the **t** switch.

During the binary spit clustering, a new cluster representative is derived from an existing representative by adding a perturbation to the existing representative. The perturbation is a fraction of the existing representative values. This fraction is specified by the **e** switch.

If during the binary slit clustering a cluster is found which contains less than F frames in the segments contained in it, it is no longer considered for further splitting. This minimum number of frames is specified

by the **m** switch. If a cluster with less that F frames is found during the K-means clustering step, the cluster is removed from the cluster inventory and the segments that were contained in this cluster are reassigned to the closest cluster (i.e. smallest distance to the cluster representative). If the verbose option (**v** option) is turned on, the cluster with too little frames are identified in the inventory listing by an asterisks at the start of a line.

The clustering process can be started with an initial inventory that is to be split further by specifying the initial model inventory by a text file which contains a list of model names through the **I** switch. The models are assumed to be found in the directory, specified by the **J** switch.

The **f** sets the minimum allowable variance for any dimension of any model. If the estimated variance of a particular model is smaller than this value, it's value is replaced with this minimum.

3.4 The re-segmentation program: vitsegm

Executing vitsegm with no command line arguments gives the following output:

```
USAGE: vitsegm [options] modellist filenames...

 Option                                        Default

 -b f    Beamwidth in loglikelihood            100
 -i f    Segment insertion cost                -22
 -v      Verbose                               off
 -w f    Parameterisation window width (ms)    25.6
 -B s    Set bigram stats input file to s      none
 -C      Ignore bigram probabilities           off
 -D      Ignore duration probabilities         off
 -F fmt  Set data file format to fmt           HTK
 -H s    Set fixed boundary desc. file dir     none
 -I s    Set fixed boundary desc. file ext     fix
 -J s    Set models directory to s             current
 -O s    Set Viterbi segment. output dir       current
 -P s    Set Viterbi segment. file ext.        vit
 -S f    Set script file to f                  none
 -X s    Set segmentation input dir            current
 -Y s    Set segmentation input extension      alab
 -Z fmt  Set segmentation input fileformat     HTK
```

A text file containing the names of the unit inventory is to be specified first, the feature vector representation files that are to be re-segmented after that. The **S** allows the expansion of the command line as described under the description of the *asegm* program.

The program assumes to be able to find a previous segmentation of the utterance that is to be re-segmented in a file with the same core name and extension which is specified by the **Y** switch in a directory specified by the **X** switch. If the program is unable to open such a file, it exits.

The resulting segmentation label files are written to the directory specified by the **O** switch and will have the same core name as the feature vector representation file with an extension specified by the **P** switch.

Boundary constraints can be specified in a file with the same core name as the feature vector representation file in the directory specified by the **H** switch. If this switch is not set, it's assumed that no boundary constraints will be specified. The extension of such a boundary constraint file is specified by the **I** switch.

The **B** switch allows the user to define a file containing the bigram probabilities that are to be used. If no such file is specified, unigram probabilities are used. The unigram probabilities are estimated from

the relative frequencies found in the *RegModelFile* format files. If a bigram probability file is specified, the beam-width, used in the Viterbi search in log-likelihood can be specified through the **b** switch.

Three switches influence the likelihood computation, used in the dynamic programming loop. Setting the **C** switch indicates that the bigram (or, if no bigram probability file is specified, unigram) probabilities are to be ignored. The **D** switch will cause the duration probability to be ignored. Finally, to control the resulting number of segments, an additive segment insertion penalty can be specified through the **i** switch.

The HTK label file format has a field for the "score" of a segment. The output of the *vitsegm* program set the values of this score field of the segments. The score only includes the log-likelihood of the frames, given the Gaussian model. These scores do not include the log-likelihood of the duration model. Bigram or unigram and insertion cost are also not included in these scores.

3.5 The re-clustering preogram: reclust

Executing reclust with no command line arguments gives the following output:

```
USAGE: reclust [options] ModelList labelfiles...

 Option                                        Default

 -f f     Set variance floor to f              1e-05
 -m i     Set training frame minimum           25
 -v       Verbose                              off
 -w f     Parameterisation window width (ms)   25.6
 -B s     Set output bigram file to s          bigram.out
 -F fmt   Set label file format to fmt         HTK
 -J s     Set models input dir to s            current
 -O s     Set models output dir to s           current
 -S f     Set script file to f                 none
 -X s     Set data file directory              current
 -Y s     Set data file extension              cep
 -Z fmt   Set parameter fileformat             HTK
```

A text file containing the names of the previous unit inventory is to be specified first, the label files specifying the new segmentation are to be specified after that. The **S** allows the expansion of the command line as described under the description of the *asegm* program.

The feature vector representation file, corresponding to each specified label file is expected to be found in the directory specified by the **X** switch. The core of the file names of these feature vector representation files are expected to the same as the label files, the extension is expected to as specified by the **Y** switch.

The previous models, are expected to be found in the directory specified by the **J** switch. The newly estimated models are written to the directory specified by the **O** switch.

The bigram file is written to the file specified through the **B** switch.

The minimum number of frames, necessary to be able to estimate a new model is specified by the **m** switch. If in the new segmentation, a certain model does not contain enough data for re-estimation, the model is removed from the inventory. A segment that was labeled as that unit is reassigned to the unit for which the likelihood of the segment is larger than any other unit in the new inventory.

The **f** sets the minimum allowable variance for any dimension of any model as the **f** switch in the *km-las* program.

3.6 ASSM utilities

There are 7 utility programs in the ASSM package which allow the user to investigate the results from the core programs. A brief description of these utilities are given below. The use of these programs is mostly self-explanatory.

- **lmdl:** Allows the user to look at the contents of a *SegFile* format file or *RegModelFile* format file. The file-type must be specified by the **F** switch.
- **mdl2cep:** Allows the user to generate a feature vector representation file of an utterance represented by a *SegFile* format file. The resulting feature vector representation file will have the number of segments specified in the *SegFile* format file. Each segment will have the specified length and within each segment, each dimension of the the feature vectors will follow the mean trajectory.
- **seglab2cep:** As **mdl2cep** except that the number of segments and segment lengths are determined by a label file. The feature vectors within a segment will follow the mean trajectory of the model corresponding to the label of the segment.
- **lenstat:** Computes the statistics of segment lengths for the segments found in a number of *SegFile* format files.
- **lhcomp:** Computes the summed log-likelihood from a number of label files which have log-likelihood scores specified for each segment.
- **viewer:** Tool to look at a grey-scale representation of the contents of a feature vector representation file under X-windows. A feature vector representation file followed by a label file are to be specified on the command line. For each dimension, the available gray-scale values are used for the dynamic range of that dimension within the utterance (therefore, gray-scale values are not comparable between dimensions). The boundaries found in the label file are indicated in the gray-scale plot as red lines.
- **segclass:** Tool to classify segments. This program takes a model inventory (in the form of a collection of *RegModelFile* format files) and feature vector representation files as input. The feature vector representation files are then classified as belonging to the class in the model inventory that results in the highest likelihood.

4 Lexicon building: LEXBLD

The *LEXBLD* package implements the lexicon building algorithm described in TR-IT-0147. It consists of 1 program which, on termination creates a file containing the lexicon consisting of all unique single words found in the training data as well as a number of multi-word entries. The following subsection describes the file formats used in this package, the following subsection describes the use of the program to build the lexicon.

4.1 File formats

The lexicon building program takes label files as input and produces a lexicon file and boundary constraint label files as output. The default format for the label files is the HTK format but other file formats are supported. Please look in the HTK documentation for a detail description of the supported label file formats. All functionality found in the HTK package with respect to input and output of label files, also applies to this program.

The lexicon file produced by the lexicon building program is in ascii form. For each lexical entry in the lexicon, a line using the following format is declared:

```
"<lexical entry>" freq score
```

The field `<lexical entry>` is a lexical entry of possibly more than one word. To clearly indicate the start and end of the lexical entry, it is embraced by " characters. The `freq` field indicates the frequency of the lexical entry in the training data. The `score` field specifies the score (used in the lexicon building algorithm) of the lexical entry. As single word entries do not have a score in the algorithm, the score field is ignored. The lexicon building program will put a 0 in this field. An example of a possible line in the lexicon file is:

```
"in the" 478 824.532
```

which would indicate that the lexical entry `"in the"` occurs 478 times in the trainingdata and has a score of 824.532.

4.2 The lexicon building program: lexbuild

The lexicon building program takes as input a number of label files specifying where acoustic unit boundaries are located. It also takes as input, label files specifying where word boundaries are located. An ascii file containing the lexical entries in the lexicon as well as information on frequency and scores (as used in the lexicon building algorithm) is produced as output.

If the *lexbuild* program is executed without any command line parameters, the following output is produced:

```
USAGE: lexbuild [options] filenames...

Option                                          Default

-v       Verbose                                off
-A s     Set auto-segment. labelfile dir        current
-B s     Set auto-segment. labelfile ext        vit
-F f     Set max time to fixed boundary         10 ms
-H s     Set fixed boundary label file dir      current
-J s     Set fixed boundary label file ext      fix
-M i     Set the lexicon size to i              1000
-O s     Set lexicon output file to s           lexicon.out
-S f     Set script file to f                   none
-W s     Set word labelfile format              HTK
-X s     Set auto-segment. labelfile format     HTK
```

After the options, a list of label files is to be specified. These label files define the position of word boundaries. The program assumes that for each word-label file, a label file containing the positions of the boundaries between acoustic labels in the corresponding utterance can be found in the directory specified by the **A** switch. The core part of the name of such an automatic-segmentation label file is assumed to be equal to the core part of the word-label file. The extension of the automatic-segmentation label file is specified by the **B** switch. If such a file can not be found, the program exits.

The desired resulting lexicon size is specified by the **M** switch. The boundary constraint files, necessary to prune the lexicon down to the desired size are written in the directory specified by the **H** switch. The boundary constraints for an utterance are placed in a file whose filename's core part is equal to the utterance file name with an extension as specified by the **J** switch.

The threshold for considering a word boundary to be crossed by an automatic segment can be specified by the **F** switch.

The **S** switch allows the user to specify a text file which contains a list of filenames of files to be processed. The list of filenames is appended to the command line and is available to avoid problems with command lines extending over the maximum size allowed by the operating system.

5 Pronunciation modeling: CMPM

The *CMPM* package implements some of the ideas described in TR-IT-0147. This package is still very much under development and should therefore be used with caution. Currently available are 4 programs. The 2 main programs are *ssm_lin_init* and *csrec*. The *ssm_lin_init* program generates statistics files for the mapping of acoustic units to phones or diphones. The *csrec* program is a segment model Viterbi decoder for recognition of a string of acoustic units given a lexicon and statistic files that correspond to the phones found in the lexicon. The other 2 programs are *binconvlex* and *vstat*. The *vstat* program is used to see the contents of statistics file generated by *ssm_lin_init*. The *binconvlex* is used to convert a lexicon file and pronunciation dictionary into a binary, fast loadable format. In the following subsection, the used file formats will be described, the subsequent subsections will describe the programs found in the package.

5.1 File formats

All label files that are used in this package are by default assumed to be in HTK format but other file formats are supported. Please look in the HTK documentation for a detail description of the supported label file formats. All functionality found in the HTK package with respect to input and output of label files, also applies to this program.

The *ssm_lin_init* program will map the acoustic label sequence to a phone-label sequence by a simple linear mapping. The program will then gather statistics of this mapping so that the parameters of the pronunciation can be estimated. The pronunciation model will estimate the pronunciation given a phonemic baseform found in a pronunciation dictionary. As it is possible that allophones occur in the transcriptions, it might be necessary to preprocess the phone transcription so that only phoneme labels occur before the mapping is performed. The preprocessing is defined by a number of preprocessing rules. The preprocessing rule file has one line per rule and each line has the following format:

```
<tgt> : [<lcontext>+]<tgt>[+<rcontext>]=<repl> ...
```

The `tgt` string identifies the phone label the rule applies to. The `lcontext` and `rcontext` are optional and specify that that replacement rule should be applied if the `tgt` phone in that context. The `repl` string is the new phone label for an occurrence of the `tgt` phone in the given context. More that one replacement rule can be specified on a line.

The `repl` label can be a special character - to indicate that the word in which the `tgt` label occurs should be looked up in a pronunciation dictionary and should be replaced by the phone label it finds in this dictionary. Two examples of rules are:

```
tcl   : tcl+t=t tcl=t tcl+ch=ch
dx    : dx=-
```

The first rule indicates that the phone label `tcl` followed by the label `t` should be replaced by the label `t`, `tcl followed by a ch should be replaced by ch and tcl should be replaced by t in all other cases. The second rule indicates that if the label dx is encountered, the word it occures in should be looked up in the pronunciation dictionary and the replacement for the dx label will be the phoneme label found in the transcription of the word in the pronunciation dictionary.`

`The lexicon and pronunciation dictionary are converted into a fast loadable binary format named` *LexBFPronDat*. `This file format contains the following information:`

- `the number of phones found in the pronunciations of the the lexical entries`
- `the phone labels`
- `the size of the lexicon`

- the lexical entries
- the pronunciation lengths (i.e. the number of phonemes) in each of the lexical entries
- the pronunciations of the lexical entries
- the frequency of the lexical entries in the training data
- the total count of all lexical entries in the training data

The mapping statistics files produced by the *ssm_lin_init* program are in a format named *BaseFormStatDat*. These files contain the following information:

- the inventory size of acoustic units found in the training data
- the labels of the acoustic units found in the training data
- the minimum and maximum length of a sequence, mapped to a phone
- the total number of mappings (equals the number of phones) in the training data
- the total number of unique acoustic unit sequences found among the sequences
- the number of unique acoustic unit sequences specified by sequence length
- the acoustic unit sequences and their frequency
- the inventory size of acoustic units found among 2-length sequences
- the inventory size of acoustic units found among $\geq$3-length sequences
- the acoustic unit labels of the units found among 2-length sequences
- the acoustic unit labels of the units found among $\geq$3-length sequences

The *csrec* program takes a decoding task description file which has the following format:

```
LEXICONFILE={<lexfile>}
LANGUAGEMODEL={<langmodel>}
INTERLEXICONUNITS={[<ilunit>,...]}
AUXUNITS={[<sunit>], [<eunit>]}
```

The lexfile field specifies the *LexBFPronDat* format file that specifies the lexicon and pronunciation dictionary for the task. The langmodel will in the future be used to specify the type of language model that is to be used for the task. Currently it can only take one value UGRAM specifying a unigram language model. The unigram probabilities are estimated from the wordfrequencies found in the *LexBFPronDat* format file, specified by the lexfile field. In the future, other language model types are to be implemented. One could think of a specification of a bigram or trigram language model type where the field would then be used to specify this modeling type and reference a certain file containing the probabilities of such a model. The ilunit fields specify the phone labels that are possible inserted in between lexical entries such as inter-word silences. The sunit and eunit fields are optional and specify the utterance start and utterance end units respectively.

5.2 The linear initialization program: ssm_lin_init

The *ssm_lin_init* program takes as input a number of label files for acoustic unit transcriptions and phone unit transcriptions. It produces *BaseFormStatDat* format files for each of the phone labels as output. The mapping from acoustic units to phone units is done by a simple time allignment. The next avaiable acoustic unit in the transcription is mapped to the next phone unit in the transcription as long as the end of the acoustic unit is before the end of the phone unit.

If the *ssm_lin_init* program is executed without any command line parameters, the following output is produced:

```
USAGE: shmm_lin_init [options] filenames...

 Option                                    Default

 -b       Baseform statistics output file  phonestat.bfs
 -v       Verbose                          off
 -A s     Set auto-segment. labelfile dir  current
 -B s     Set auto-segment. labelfile ext  vit
 -L s1 s2 Label prep.: s1 cmnds, s2 lexicon off
 -M s     Set the word labelfile dir. to s current
 -N s     Set the word labelfile ext. to s wrd
 -O s     Set baseform statistics dir to s current
 -D       Compute diphone statistics       off
 -S s     Set script file to s             none
 -W s     Set baseform labelfile format    HTK
 -X s     Set auto-segment. labelfile format HTK
 -Y s     Set word labelfile format        HTK
```

After the options, a list of label files is to be specified. These label files define the position of phone label boundaries. The program assumes that for each phone-label file, a label file containing the positions of the boundaries between acoustic labels in the corresponding utterance can be found in the directory specified by the **A** switch. The core part of the name of such an automatic-segmentation label file is assumed to be equal to the core part of the phone-label file. The extension of the automatic-segmentation label file is specified by the **B** switch. If such a file can not be found, the program exits.

The **S** switch allows the user to specify a text file which contains a list of filenames of phone label files to be processed. The list of filenames is appended to the command line and is available to avoid problems with command lines extending over the maximum size allowed by the operating system.

The phone label files can be preprocessed by a set of rules. The rules should be specified in a format as described in the previous section. The file containing the preprocessing rules can be specified through the **L** switch. This also requires a *LexBFPronDat* format file to be specified. If a preprocessing rule is used that requires that the pronunciation of a word has to be looked up in a pronunciation dictionary, this *LexBFPronDat* format file should cover the words that require to be looked up. It also requires that the word label files corresponding to the phone-label files can be found in the directory specified through the **M** switch. The core part of the name of a word-label file is assumed to be equal to the core part of the correspond phone-label file. The extension of the word-label file is specified through the **N** option.

The *ssm_lin_init* program computes besides statistics on mappings from acoustic units to phone units also statistics on phone labels. These statistics are stored in a *BaseFormStatDat* format

file. The name of this file can be specified through the **b** switch.

5.3 The Viterbi decoding program: csrec

The *csrec* program takes as input a label file with the transcription of an utterance in terms of acoustic units and a task description file in the format described above. It produces a word-label file as output.

If the *csrec* program is executed without any command line parameters, the following output is produced:

```
USAGE: csrec [options] taskdescript filenames...

 Option                                    Default

-v      verbose                            off
-B s    set baseform bigram file name      phonestat.bfs
-L s    set acoustic label file format     HTK
-M s    set baseform model directory       current
-P s    set phone recognition results ext  prec
-Q s    set word recognition results ext   wrec
-R s    set the recognition results dir    current
-S f    set script file to f               none
```

After the options, the task description file is to be specified. After that, the user should give a list of labelfiles which contain the transcription in terms of acoustic units of the utterances which are to be decoded.

The complete phone inventory for the task consists of the phones found in the *LexBFPronDat* format file specified in the task description and in the task description itself. The *BaseFormStatDat* format files corresponding to this phone inventory are assumed to be found in the directory specified by the **M** switch. The *BaseFormStatDat* format files are assumed to be named as the phones in the inventory. The phone bigram statistics are assumed to be found in a *BaseFormStatDat* format file in the same directory and is assumed to be named as specified by the **B** switch. If any of these files are not found, the program exits.

The resulting word-label (and phone-label) file are written in a directory specified by the **R** switch. The core part of the file name of these result files are the same as the core part of the acoustic label file. The extensions are specified through the **P** and **Q** switch.

As with the *ssm_lin_init* program, the command line can be extend by use if the **S** switch.

5.4 CMPM utilities

There are currently 2 utilities available in the *CMPM* package. The use of these utilities is self-explanatory. A brief description of the 2:

- **binconvlex:** Converts an ascii form lexicon file as described under the *LEXBLD* package subsection and a pronunciation dictionary into a *LexBFPronDat* format file. The only supported format for the pronunciation dictionary is the format of this pronunciation dictionary in the TIMIT database.
- **vstat:** Prints the contents of a *BaseFormStatDat* format file in ascii form.