UNIVERSITY OF ZADAR

INTERDISCIPLINARY SUMMER SCHOOL, WORKSHOP AND ROUND TABLE IN COMPUTATIONAL LINGUISTICS, COGNITIVE AND INFORMATION SCIENCE

INTRODUCTION TO SPEECH RECOGNITION

Exercise in ASR using HTK

prof.dr.sc. Davor Petrinović

Branimir Dropuljić, dipl.ing.

University of Zagreb

FER

Zadar, *22nd of August - 3rd of September 2010*

Content

Introduction 1

HTK 2

System customization for the exercise 3

Task 1 5

Task 2 11

Task 3 13

Appendix 1 14

  Example of ten test sentences 14

Appendix 2 15

  Instructions for building acoustic model for English language 15

Introduction

The idea of this practical part of the course *Introduction to speech recognition* is to learn basics about hidden Markov models (HMM), to learn how to build a new acoustic model for English language using HMMs and to test the quality of this model. Model testing will be performed on utterances with limited vocabulary and strictly defined word-to-word transitions, which will be recorded during this exercise, by each student individually. Finally, quality and complexity of the acoustic model will be compared with another model, which will be constructed from all the utterances of all students recorded during these exercises.

Furthermore, students will learn how to work with the Hidden Markov Model Toolkit (HTK) which is a main tool for building and testing acoustic and language models. It provides an opportunity to build models from scratch and more importantly - step by step.

The main goal of this exercise is to teach the students to research and explore the ASR world and gain hands-on experience using their own example.

This exercise is designed for individual work but students can also work in pairs or in groups.

HTK

The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models. HTK is primarily used for speech recognition research although it has been used for numerous other applications including research into speech synthesis, character recognition and DNA sequencing. HTK is in use at hundreds of sites worldwide.

HTK consists of a set of library modules and tools available in C source form. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems. The HTK release contains extensive documentation and examples.

HTK was originally developed at the Machine Intelligence Laboratory (formerly known as the Speech Vision and Robotics Group) of the Cambridge University Engineering Department (CUED) where it has been used to build CUED's large vocabulary speech recognition systems. In 1993 Entropic Research Laboratory Inc. acquired the rights to sell HTK and the development of HTK was fully transferred to Entropic in 1995 when the Entropic Cambridge Research Laboratory Ltd was established. HTK was sold by Entropic until 1999 when Microsoft bought Entropic. Microsoft has now licensed HTK back to CUED and is providing support so that CUED can redistribute HTK and provide development support via the HTK3 web site.

HTK will serve as the main tool for the practical part of this course. Registering, downloading and preparing the HTK will be described in the next chapter. More detailed description of the HTK, as well as the HTK Book (which will be frequently referred in this exercise guide), is available in http://htk.eng.cam.ac.uk/.

The HTK Book consists of four main chapters. The first chapter called Tutorial Overview elaborates basics of the speech recognition in HTK and also gives one typical example of how to build and test an acoustic model. The second chapter called HTK in Depth deals with detail theoretical background of the whole recognition process in HTK. The Third chapter called Language Modeling is about upgrading an acoustic model to a language model. The last chapter called Reference Section consists of all the HTK script descriptions and also descriptions of all possible errors and warnings.

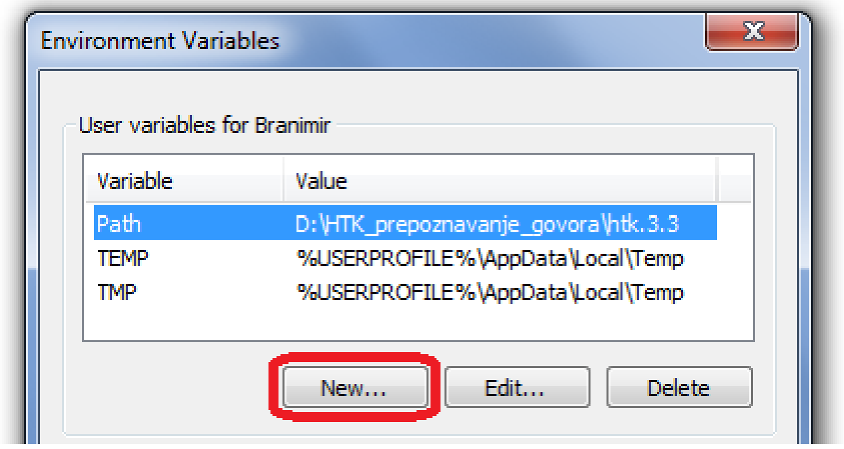
System customization for the exercise

For this practical exercise it is necessary to have a PC (desktop/laptop) with compatible audio input and output units (microphone and speakers) and with Windows or Linux operating system installed.

System customization will be described through following steps:

1. Check the compatibility between your PC and audio units.
2. Register on the HTK site to get free access to the HTK source code and the HTK Book.
   - Link for registration: http://htk.eng.cam.ac.uk/register.shtml.
3. Login and download a current version of the HTK source code (3.4.1) according to your OS.
   - Link: http://htk.eng.cam.ac.uk/download.shtml.
4. Download the HTK Book: http://htk.eng.cam.ac.uk/download.shtml.
5. If dynamic programming language Perl is already installed on your PC and working properly go to the next step, otherwise download it and install it from http://www.activestate.com/activeperl/downloads according to your OS, because some HTK scripts run several Perl scripts in background.
6. Audacity is useful free software for recording and editing speech utterances that can be used for this exercise. It can be freely downloaded from http://audacity.sourceforge.net/, according to your OS.
7. Before beginning with the exercise, HTK needs to be installed on your PC:
   - Installation process for Windows:

     i. Set the path to your HTK folder with the HTK scripts. To do this, go to *`Control_Panel\System\Advanced_Settings\`* *`\Environment_Variables`* and add the new variable named *`Path`* with the path of a folder with the HTK scripts as a value (Fig. 1).



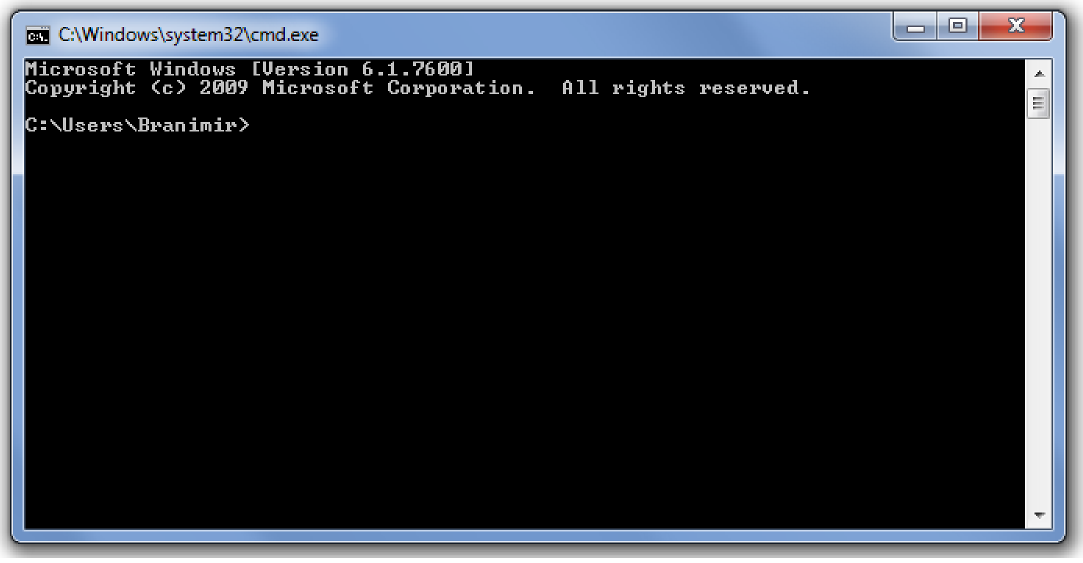
Variable	Value
Path	D:\HTK_prepoznavanje_govora\htk.3.3
TEMP	%USERPROFILE%\AppData\Local\Temp
TMP	%USERPROFILE%\AppData\Local\Temp

Fig. 1. Example of setting variable Path in Windows 7

- Installation process for Linux:
    i. The instructions are available at http://htk.eng.cam.ac.uk/docs/inst-nix.shtml.

When you finish all this steps, you are ready to start working on this exercise through following tasks.

It should be noted that all HTK scripts in this exercise should be run in the command prompt (*cmd*) of your PC (Fig. 2).



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Branimir>
```

Fig. 2. Example of setting variable Path in Windows 7

Task 1

Test the accuracy and correctness of the trained acoustic model of English language, built from five hundred utterances which are spoken by ten non-native speakers. Acoustic model was built and prepared earlier and the results are given in an appropriate form. Perform testing using a dictionary with limited vocabulary and strictly defined word network. Example of input prompts in Appendix 1 should be used to create a grammar and then to prepare ten random sentences that comply with this grammar. Corresponding dictionary and word network must be built for the same example. Finally ten utterances that correspond to the ten generated random prompts must be to record for testing.

First step is to build the word network for this particular experiment from sentences in Appendix 1. First you have to define the task grammar, which is a simple .txt file and defines possible word to word transitions in test utterances. To do this, you need to study HTK Book, especially chapters 3.1.1 and 12.3.

It is important to mention that sentences from Appendix 1 are not the only possibilities defined by grammar. Hence, to enable defining strict grammar a few rules will be described:

- Sentence can have seven or eight words (nine or ten if you include words SENT-START and SENT-END).
- Each word in a sentence (except the last two words – when SENT-END is excluded) is randomly selected according to its neighboring words and between maximally three options. Also, one word has an optional appearing, which is random as well.
    - o The last word in a sentence (SENT-END excluded) depends on its previous neighbor. All possible options are specified through given sentences.

Sketch the grammar in the provided space:

Now you can create the word network using this grammar.

Notice: Beside the two mentioned sections (3.1.1 and 12.3) that explain this first step, further insight into each of HTK functions can be found in the Reference Section, chapter 4 of the HTK Book.

Second step is to build the dictionary which will be used in the word recognition process.

To understand the speech recognition process in details complete chapter 1 of the HTK Book needs to be studied. Here we can say that there are two main ways to perform speech recognition:

- isolated word recognition
- continuous speech recognition

In this task and also in the whole exercise we will deal with continuous recognition which is more complicated, but better suited for natural pronunciation. Among other things this means that recognition will be performed in three levels (see chapter 13.1 of the HTK Book for more details). At the first level, only sub-word models exist. This means that each word is represented by a chain of sub-word models. Such approach is suitable for unlimited vocabulary speech recognition, since new words can be modeled using sub-word models. Here we can decide: we can create only monophone models, only triphone models or combination of monophone, biphone and triphone models. Word transcriptions for these three methods are given in Table 1.

Table 1. Examples of word transcription

Sentence	Transcription using models	Type of transcription
Dog eats cat.	**sil d oh g sp iy t s sp k ae t sil**	Monophone
	sil sil-**d**+oh d-**oh**+g oh-**g**+iy **sp** g-**iy**+t iy-**t**+s t-**s**+k **sp** s-**k**+ae k-**ae**+t ae-**t**+sil **sil**	Triphone (cross-word, C-W)
	sil d+oh d-**oh**+g oh-**g sp iy**+t iy-**t**+s t-**s sp k**+ae k-**ae**+t ae-**t sil**	MBT Combination (word-internal, W-I)

The 'sil' model represents a longer pause at the beginning and at the end of each sentence, whilst a short pause is represented using a 'sp' model. The other models are, for example 'sil-d+oh', 'd-oh+g', 'd+oh', etc. At this moment transcription using models may seem confusing. So for now we can only say that in our exercise we'll used combined, word-internal transcription.

Try to analyze a little bit more the word-internal (W-I) transcription example; are there any monophone models in this example? If yes, write them in the provided space; if no, try to figure out the example which includes monophones (use *dictionary_beep.txt* if you need help) using W-I transcription and write it together with the transcription:

At the other two higher levels, sub-word models (HMMs) are combined into a word network. The relationship between words in vocabulary for recognition and their transcription with HMM models is given in a file called dictionary. To learn how to create the dictionary read chapters 3.1.2 and 12.7 of the HTK Book. Notice that vocabulary for this task is very limited, only thirteen words (fifteen with SENT-START and SENT-END included) appearing. Although W-I transcription is used, the dictionary should be constructed using monophone transcription. Conversion from one way of transcription to another is performed automatically during the recognition phase (see chapter 3.4.1 and 12.8 of the HTK Book).

Now you can finally construct the dictionary.

Tips:

- Use the example prompts (*prompts_teExmpl.txt*) to create a word list that is required for building the dictionary.
  - You can find perl script *prompts2wlist* and example prompts in your main folder. Example of running in cmd window:
    - `>> perl prompt2wlist prompts_teExmpl.txt wordlist_te.txt`
- After creating the wordlist, it is necessary to add words SENT-START and SENT-END in it.
- You can find the British English Pronouncing Dictionary (*dictionary_beep.txt*) with nearly 250000 words and with SENT-START and SENT-END added in your main folder.
  - Don't remove *global.ded* script from your main folder because the script for building the dictionary (*HDMan*) uses it.
- Add square brackets before 'sil' transcription for SENT-START and SENT-END in yours dictionary.

Third step is to record 10 random utterances complying to the designed grammar that are similar to the example in Appendix 1 and to parameterize them into sequences of feature vectors that are used as observations in HMM models.

In order to prepare 10 random prompts that will be used in this experiment, a special tool can be used that is explained in chapter 12.6 of the HTK Book. This file will also be used for evaluation of recognition accuracy.

It is recommended to use Audacity for recording the utterances, although any other similar sound recorder can be used as well. **Set the parameters to 16 kHz sampling frequency and 16 bits, mono, per sample (*Edit\Preferences\ \Devices* and *Edit\Preferences\Quality*).** These settings are very important, since the speech coding tool (*HCopy*) that is used for preparation of feature vectors for ASR expect exactly this recording format. Save the files in .wav format using the Export option (*File\Export*, see Fig. 3).

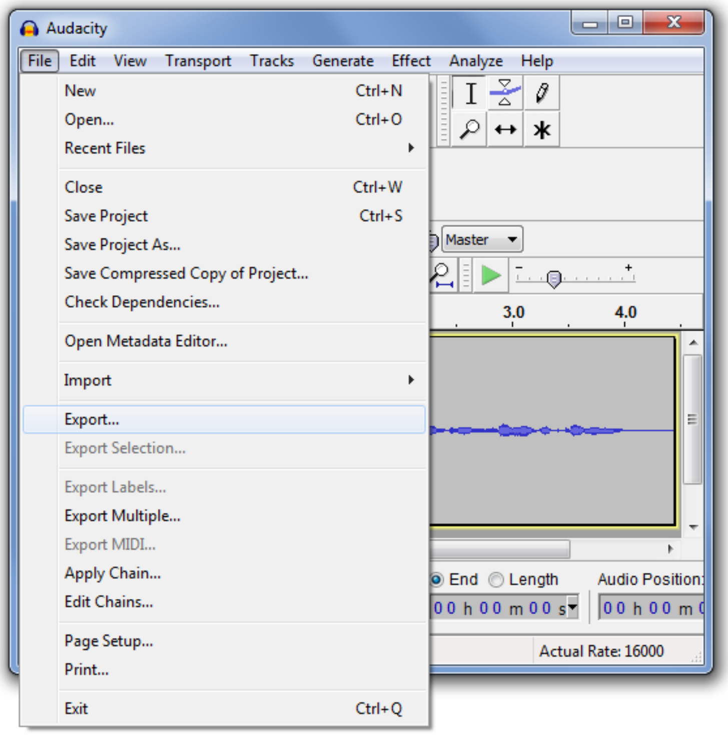


Fig. 3. Example of exporting file in .wav format in Audacity 1.3.12 (Beta)

Save each utterance in separate wav file (T01.wav, T02.wav, etc.). Play the recorded utterance before saving to check if utterance is consistent with the sentence in prepared prompt file. Check also if any unwanted loud noise is added to the utterance, for example other voices apart from your own voice. If so, repeat the recording process for this sentence for better results when testing. Also, if the recorded utterance contains prolonged pauses (e.g. silence periods at the beginning or at the end of the utterance), that are longer than one second, cut them in Audacity before saving.

After recording, convert the utterances into the MFCC input vector stream that will be used as an input for the HTK. Use the same coding parameters as described in the chapter 3.1.5 of the HTK Book with one exception: set the TARGETKIND to MFCC_0_D_A. This means that one vector has 12 MFCC coefficients + 1 zero coefficient (0) + 13 delta coef. (D) + 13 acceleration coef. (A). Hence, the total number of coefficients per vector is 39. You can read chapter 5 of the HTK Book if you want to explore details about coding options in HTK.

Try to calculate how many vectors will be generated as an input for utterance that is 5 seconds long (see the Fig 5.2 of the HTK Book) and write the procedure in the provided space:

Tips:

- Example of creating the random prompt file:
  - `>> HSGen -s -n 10 wordnet dict_te > prompts_te.txt`
- Adapt *prompts_te.txt* to the appropriate format for HTK (use *prompt_teExmpl.txt* as an example).
- Use *hcopy_te.conf* from the main folder with all parameters set.

Fourth step is to run the speech recognition process for the recorded utterances using HTK and to test the accuracy and the correctness of the supplied acoustic model for the English language.

As it is mentioned earlier, recognition will be performed using the supplied acoustic model that was built for the English language from utterances of ten non-native speakers (five men and five women at the age between 20 and 25). Each person uttered fifty sentences which were taken randomly from the Internet. All sentences were manually reviewed and adopted for this purpose. The vocabulary for this model is obviously unlimited (actually it is only limited by the size of the BEEP dictionary that contains approximately 250000 words). HMM models are combined from monophones, biphones and triphones (W-I transcription).

Acoustic model that are contained in this HMM model will be described in more details in the next Task.

Read chapters 3.4.1 and 13.3 of the HTK Book to learn how to perform recognition process. Pay attention to the variables FORCECXTEXP and ALLOWXWRDEXP (see chapter 12.8 of the HTK Book if you are interested in more details). Combine parameter values for recognition (*t* – pruning, *p* – word insertion penalty and the *s* – grammar scale factor) and observe changes in the result (see also chapters 13.1 and 13.2 of the HTK Book if necessary).

After finishing recognition, you can finally perform comparison between output result from testing process with the reference sentences from *prompts_te.txt*. See chapters 3.4.1 and 13.4 of the HTK Book.

Try to locate all type of errors (deletions, substitutions and insertions) in each sentence. Explain where in a sentence can each type of error appear? Write your answer in the provided space (if it's difficult to describe the location use some examples to describe it):

Finally, write overall recognition results (output from *HResults*) for the sentence level and for the word level as well in the provided space:

Tips:

- Use *hvite_te.conf* from the main folder with parameters adjusted as needed.
- Use acoustic model *hmmdefs_T1.mmf* together with *macros_T1* and also list of all hmm models included in acoustic model (*tiedlist_T1*) from the main folder.
- Use the perl script *prompts2mlf* from the main folder.
- Use the –*t* option when running comparison analysis (*HResults*) to printout the time-aligned word transcription of the reference according to the recognition result for each sentence.

Task 2

Perform automatic training of the acoustic model that was used in the previous Task from 500 speech utterances. You can use directions written in Appendix 2 to speed up the procedure which consist of a many complex training steps. The focus of this experiment is to gain understanding of each step in this training procedure. Therefore you should also read the selected chapters of the HTK Book in parallel.

The procedure of building and training acoustic model of the English language will be described here in short steps. As mentioned earlier, ten non-native speakers, five men and five women took part in recording five hundred utterances. Each person uttered fifty sentences which were taken randomly from the Internet. All sentences were transcribed with monophone HMM models at first. After a few iterations of parameter re-estimation, biphone and triphone models were added to the acoustic model and re-estimation was repeated several times (using W-I transcription). Since the vocabulary is virtually unlimited, while used training speech material is not big enough to cover all possible triphone models of the english language, state tying of HMM model was performed as the final step of the training procedure.

Follow the instructions in the Appendix 2 for creating acoustic model used in the previous Task. Study chapters 3.1.2, 3.1.4, 3.1.5, 3.2 and 3.3 of the HTK Book trying to find the ground why particular instructions were chosen.

Finally, try to answer the following questions:

1. Which method is used to initialize the monophone HMM models? Describe the method in short steps.
2. Why do we need HERest? What is happening during this process?
3. What actually happens during the tying process?
4. Why do we need the *Questions*?

Task 3

Repeat the speech recognition process for the utterances recorded in the third step of the first Task, but now with the new acoustic model, constructed from the new speech database comprised of all utterances of all students recorded during first two days. This acoustic model was built using only the words from our limited dictionary (13 words). Compare the accuracy and the correctness of the new acoustic model with the results from the Task 1.

It should be noted that the new acoustic model is constructed for the specific purpose and it should be used only for recognizing words from the grammar which was constructed in the first Task. Hence, vocabulary is limited and the model was built only from HMMs that can appear in our word network.

Perform the recognition and write the results in the provided space:

Compare the results with the one from Task 1 and discuss the differences in the provided space:

Tip:

- Use *hvite_te.conf* from the main folder with parameters adjusted as needed. Also, use acoustic model *hmmdefs_T3.mmf* together with *macros_T3* and also list of all hmm models included in the new acoustic model (*tiedlist_T3*) from the main folder.

Appendix 1

Example of ten test sentences

1. DOG EATS CAT ON A WONDERFUL SUNNY DAY.
2. CAT EATS MOUSE ON A WONDERFUL SUNNY AFTERNOON.
3. MOUSE EATS DOG ON A RAINY AFTERNOON.
4. DOG EATS DOG ON A RAINY MORNING.
5. CAT LIKES CAT ON A WONDERFUL RAINY AFTERNOON.
6. MOUSE EATS MOUSE ON A SUNNY DAY.
7. DOG LIKES MOUSE ON A WONDERFUL SUNNY DAY.
8. CAT LIKES DOG ON A SUNNY AFTERNOON.
9. MOUSE LIKES CAT ON A WONDERFUL RAINY MORNING.
10. DOG LIKES MOUSE ON A RAINY MORNING.

Appendix 2

Instructions for building acoustic model for English language

```
1. >> perl prompts2wlist prompts_tr.txt wordlist_tr.txt
```

2. Add SENT-START and SENT-END to the wordlist_tr.

```
3. >> HDMan -m -w wordlist_tr.txt -n monophones1_tr -l dlog_tr dict_tr
   dictionary_beep.txt dictionary_names.txt
```

4. Add square brackets before 'sil' transcription for SENT-START and SENT-END.

```
5. >> HCopy -T 1 -C hcopy_tr.conf -S code_tr.txt

6. >> perl prompts2mlf words.mlf prompts_tr.txt

7. >> HLEd -l * -d dict_tr -i phones0.mlf mkphones0.led words.mlf

8. >> HCompV -C hcompV.conf -f 0.01 -m -S train.txt -M hmm/hmm0 proto
```

9. Fill the files macros and hmmdefs.mmf in folder hmm/hmm0 with the parameters from vFloors and proto.

   Fill the hmmdefs.mmf with the definition of hmm models from the monophones1_tr list ('sp' model excluded).

   Use the hmm body of model proto for initial definition.

10. Make monophones0_tr (the same file as monophones1_tr, only without model 'sp').

```
11.      >> HERest -C herest.conf -I phones0.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm0/macros -H hmm/hmm0/hmmdefs.mmf -M hmm/hmm1
   monophones0_tr

12.      >> HERest -C herest.conf -I phones0.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm1/macros -H hmm/hmm1/hmmdefs.mmf -M hmm/hmm2
   monophones0_tr

13.      >> HERest -C herest.conf -I phones0.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm2/macros -H hmm/hmm2/hmmdefs.mmf -M hmm/hmm3
   monophones0_tr
```

14. Copy hmmdefs.mmf and macros to folder hmm4 and insert the 'sp' model in the hmmdefs.mmf, transition matrix for the 'sp' model is:

'0.000000e+000 1.000000e+000 0.000000e+000'

'0.000000e+000 6.000000e-001 4.000000e-001'

'0.000000e+000 0.000000e+000 0.000000e+000'

```
15.      >> HHEd -H hmm/hmm4/macros -H hmm/hmm4/hmmdefs.mmf -M hmm/hmm5
   sil.hed monophones1_tr

16.      >> HLEd -l * -d dict_tr -i phones1.mlf mkphones1.led words.mlf

17.      >> HERest -C herest.conf -I phones1.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm5/macros -H hmm/hmm5/hmmdefs.mmf -M hmm/hmm6
   monophones1_tr

18.      >> HERest -C herest.conf -I phones1.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm6/macros -H hmm/hmm6/hmmdefs.mmf -M hmm/hmm7
   monophones1_tr
```

19. Add word 'silence' with the transcription 'sil' at the end of the dict_tr file (notice that the word 'silence' is written with lower cases).

```
20.      >> HVite -l * -o SWT -b silence -C hvite_tr.conf -a -H
   hmm/hmm7/macros -H hmm/hmm7/hmmdefs.mmf -i aligned.mlf -m -t 250.0 -y
   lab -I words.mlf -S train.txt dict_tr monophones1_tr

21.      >> HERest -C herest.conf -I aligned.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm7/macros -H hmm/hmm7/hmmdefs.mmf -M hmm/hmm8
   monophones1_tr

22.      >> HERest -C herest.conf -I aligned.mlf -t 250.0 150.0 1000.0 -
   S train.txt -H hmm/hmm8/macros -H hmm/hmm8/hmmdefs.mmf -M hmm/hmm9
   monophones1_tr

23.      >> HLEd -n triphones1_tr -l * -i wintri.mlf mktri.led
   aligned.mlf

24.      >> HHEd -H hmm/hmm9/macros -H hmm/hmm9/hmmdefs.mmf -M hmm/hmm10
   mktri.hed monophones1_tr

25.      >> HERest -C herest.conf -I wintri.mlf -t 250.0 150.0 1000.0 -S
   train.txt -H hmm/hmm10/macros -H hmm/hmm10/hmmdefs.mmf -M hmm/hmm11
   triphones1_tr
```

```
26.      >> HERest -C herest.conf -I wintri.mlf -t 250.0 150.0 1000.0 -s
   stats -S train.txt -H hmm/hmm11/macros -H hmm/hmm11/hmmdefs.mmf -M
   hmm/hmm12 triphones1_tr

27.      >> HDMan -b sp -n fulllist_tr -g global_tri.ded -l flog
   dictionary_tri dictionary_beep.txt dictionary_names.txt
```

28. Add models 'aa','ow','ay','ao' and 'ea' at the end of the HMM list (fulllist_tr) because of the bug in the HDMan script.

```
29.      >> HHEd -H hmm/hmm12/macros -H hmm/hmm12/hmmdefs.mmf -M
   hmm/hmm13 tree.hed triphones1_tr > log

30.      >> HERest -C herest.conf -I wintri.mlf -t 250.0 150.0 1000.0 -S
   train.txt -H hmm/hmm13/macros -H hmm/hmm13/hmmdefs.mmf -M hmm/hmm14
   tiedlist_tr

31.      >> HERest -C herest.conf -I wintri.mlf -t 250.0 150.0 1000.0 -S
   train.txt -H hmm/hmm14/macros -H hmm/hmm14/hmmdefs.mmf -M hmm/hmm15
   tiedlist_tr
```