Speech Technology Project 2004
Building an HMM Speech Recogniser for Dutch

Frans Adriaans Markus Heukelom Marijn Koolen
Tom Lentz Ork de Rooij Daan Vreeswijk
Supervision: Rob van Son

9th July 2004

Contents

1 Introduction

In the Speech Technology Project of 2004, a group of students in the Master's fase of Artificial Intelligence were assigned the task of building a speech recogniser for Dutch using Hidden Markov Models (HMM's). The goal of this project was to build a robust phoneme driven recogniser. That means that it should be able to generalise both from speaker specific properties and that its training should be more than just instance based learning. In the HMM paradigm this is supposed to be the case, but we wanted to put this to practice. Also, the system should serve as a stepping stone for future projects.

As the time scope was limited (less than a month) and to be able to focus on more specific issues than Hidden Markov Modelling in general, the HTK toolkit was used. Secondly, to reduce the difficulties of the task, a very limited language model was used. Future research can be directed to more extensive language models.

This report is one of the products of the project. The other products are a HMM for Dutch spoken language (albeit for a very restricted domain) and a recipe for building HMM speech recognisers, especially for Dutch and using HTK.

The report can be read as a whole, but those interested in the theoretical issues dealt with or in the reusing the "products" can skip § 2. That section provides an overview of how the project was done in practice (project is meant as opposed to product, i.e. the system we built).

§ 3 will deal with those issues that are interesting from a theoretical point of view, for those who want to incorporate or extend the Speech Recogniser and for subsequent Speech Technology Projects, that will use, test, improve and/or alter it.

As it is tedious to keep citing the HTK's documentation, the only reference to it is here: [Cam02] is the toolkit itself and [You02] is the main source of documenation, the HTK Book.

2 Group Dynamics and Project Progress

2.1 Commitments and Planning

At the 7th of June meeting preference was given to trying to get the HTK running and read through the manuals. Some people encountered some problems running HTK, but the fact that binaries were available[1] solved that.

At the 14th of June meeting, all participants felt they knew enough about the HTK in general to get started. Tasks were dispatched; for every task, one person is responsible, while others might work on it as well. The task division and ther people assigned to the tasks at that moment are:

Recording the sentences Marijn (responsible), Frans, Ork. Anyone with spare time will assist as well.

Dictionary (completion) Ork

Documentation Tom (resp.), Daan

Convert Praat labels to phonetic MLF Tom (resp.), Markus

Training HMM's Markus (resp.), Daan

Installing HTK on linux "OW" machines (renderfarm) Markus (resp.), Ork, Daan

Coordination Ork

For pragmatic reasons, the project was restricted to the same domain as the HTK tutorial suggests, namely instructions that a telephone can perform, like "Dial one two zero five" or "Dial X", where X is the (full, first or family) name of one of the team members.

2.2 Proceedings of the Project

The first main issue encountered was collecting data. Data in this project means *spelling–pronunciation*-pairings (per word), in HTK terminology and in the rest of this report named a dictionary.

Rob van Son provided a quite large wordlist (graphemic) and 1000 recorded sentences (as WAV-file) with corresponding phonetic transcriptions (called the IFA[2] corpus). He provided a dictionary as well.

[1]but a bit hidden in the HTK website

[2]Instituut voor Fonetiek Amsterdam, i.e. Insitute of Phonetic Sciences Amsterdam

2.2.1 Unifying data formats in accordance with HTK standards

One of the most important concerns was the data formats. The work in the first week concentrated on defining usable formats within the limits of HTK and translating all data into those formats.

These solutions are incorporated in the final shell script, that generates an HMM speech recogniser from the data. This was mainly done to ensure that others trying to do similar work can disregard these issues if they wish to do so[3].

2.2.2 Creating data sets

The IFA corpus is not the only data set used. This set was of course domain-unrelated. Therefore on the 17th of June the creation of a substantive amount of domain-related data for both training and testing purposes was initiated. The sentences (as text) were generated using the grammar. There were 150 "sentences" (see appendix A); Frans, Markus, Ork and Tom read out these sentences in the silent recording studio at the Instituut voor Fonetiek (Institute of Phonetic Sciences, abbreviated to IFA).

2.2.3 Finishing the HMM generator and testing its performance

The HTK tools create a series of HMM, each one a bit more sophisticated or trained better on the data. After creating a HMM with triphone states, the data becomes too sparse (just as indicated in [You02]), so these need to be clustered according to some shared properties.

To coordinate all the solutions into one system, a `compile` script was built that could handle all the steps in the tutorial. The idea of this script is that it should be possible to run the entire tutorial on any machine with HTK installed with only a few prewritten scripts and config files. This script allowed both rapid debugging and easy retracing if a bug was introduced. This last thing was important, while debugging or extending the system, to avoid messing up the whole system and not being able to repair the error anymore. The script called for execution of various other scripts. See the next section for more details.

2.2.4 Monophone list rewriting

In the days between the 23th and the 30th of June, a very stubborn bug stopped the HMM generation short. At the point of building triphone mod-

[3]Of course, those interested in these settings are invited to inspect the script and add adaptations.

els, the HTK did not recognise its own output anymore. The clustering tool found only two clusters and gave a lot of warnings, and the tools that were to be used with the clusters had fatal errors. This resulted from the fact that the file `monohpones_all`, generated from all the dictionaries, contained some entries that were not phonemes in the SAMPA alphabet. This problem probably originates from typing errors in one of the dictionaries. These entries (`: * newline`) were removed manually before re-running from step 4.

2.2.5 SP state

The `sp` state[4] caused some confusion, especially with the usage of `monophones0` and `monophones1` and in knowing where to have the sp state, and where not. This caused problems during the testing; the current `compile` script correctly adds the `sp` state where needed.

2.2.6 Unused dictionary entries

Entries in the dictionary (the small one generated from the toy grammar) which were not pronounced caused crashes in the testing stage. For example, we had the name Tom transcribed as `t o m` and
t O m. The `HVite` pass at step 8 descided that only the second version (`t O m`) was pronounced, so the first version was never placed in the triphone mlf files. During the testing stage `HVite` tries to find an entry in the HMM for that pronounciation, which is not there. See the following command line session:

```
-> HVite -S test.scp -H hmm15/macros -H hmm15/hmmdefs -l '*' \
 -i recout.mlf -w wdnet -p 0.0 -s 5.0 -T 7 dict tiedlist -C \
in/config.hconfv

Read 2821 physical / 6702 logical HMMs
Read lattice with 605 nodes / 1106 arcs
  ERROR [+8231]  GetHCIModel: Cannot find hmm [o-]m[+???]
 FATAL ERROR - Terminating program HVite
```

Removing the `TOM t o m sp` entry from the dictionary solved this problem. This had to be done for the words *Tom*, *Markus*, *Heukelom* and *nummer*. Markus Heukelom had to be removed entirely because his name was accidentally omitted from the testprompts file. As a result of this his name

[4]short pause

never got recorded, and it contained a few unique triphones which resulted in HVite not being able to reconstruct his name from other data.

Meanwhile, a language model was prepared. Language modelling was the last stage in the project due to time constraints. The most frequent words in the speech fragments of the IFA corpus were extracted and a bigram model was made for these words. The words were ranked according to frequency and then this list was pruned after the words that account for a certain percentage of the corpus. This percentage is a parameter; if it is set to 80%, for instance, then the corpus would decrease in size by 80% if all these words were deleted from it.

The other 20% of the corpus would then consist of words outside the language model.

3 Theoretical Approach

3.1 The Grammar

For the limited scope of this project, only a few words and a toy grammar were defined. The words were digits, the names of the project members and the two imperatives "draai" (dial) and "bel" (call). The grammar was defined in BN-form, as follows:

Format 3.1 $variable *defines a* phrase *as anything between the subsequent = sign and the semicolon, where | stands for a logical or, i.e.* $\vee$*. Brackets have the usual grouping function and square brackets denote optionality.*

The used toy grammar was:

```
$digit = EEN | TWEE | DRIE | VIER | VIJF |
         ZES | ZEVEN | ACHT | NEGEN | NUL;
$name  = [ ROB ] (VAN SON) |
             [ FRANS ] ADRIAANS |
             [ TOM ] LENTZ |
             [ MARIJN | MARINUS ] KOOLEN |
             [ ORK ] (DE ROOIJ) |
      [ MARKUS ] HEUKELOM |
      [ DAAN ] VREESWIJK;
( SENT-START ( DRAAI <$digit> | BEL $name) SENT-END )
```

With the HTK tool HParse a lattice file was built, which is a network containing words and transitions according to the grammar. The HSGen tool can use this lattice file to generate random sentences. See § 4.1 on page 10 for the use of these sentences.

3.2 A Pronunciation Dictionary Using SAMPA

In order to train the HMM network, a large pronunciation dictionary is needed. The format of this dictionary is dictated by HTK as follows:

Format 3.2 word [output]* pronprob* phonemes
*in which the * means that the entry is optional; the optional arguments were always included to avoid misrecognition of phonemes starting with a digit.*

SAMPA [Cha95] was used as phonetic alphabet.

This pronunciation dictionary was then used to find the phonetic transcriptions of the words in our domain.

3.2.1 Converting Words into Phonemes

NeXTeNS was used to automatically generate a pronunciation dictionary, i.e. the phonemes for approximately 1.5 milion words from a list based on Dutch newspapers[5]. This was done in multiple steps:

- The words in an already given but smaller pronunciation dictionary were deleted from the word list, so that words for which pronunciations were already available were not analyzed again. This was done for two reasons:

    1. The other dictionaries are hand corrected, and this is not. So it is better to have phonemes from those dictionaries than from this 'fallback' dictionary.
    2. The process is slow; as little words as possible should be passed through it.

- The resulting word list with about 1.3 milion words left is then inserted into NeXTeNS phoneme generation script.

The phoneme extraction process does have a few bugs: for the first few minutes, it processes words at a speed of 14.000 words per minute, and after about 150.000 words it slows down to zero. This could indicate some sort of memory leak. Increasing the heap size (with the `--heap` parameter)

[5]provided by Rob van Son

increased the maximum number of processable words to arround 230.000, but the slowdown still occured.

This was solved by restarting after every 100.000 words. The speed was reasonable after this adjustment.

3.3 Recording

In order to train and test the recognizer on the domain and on the voices of the team members, 150 sentences were automatically generated from the grammar with HTK's HSGen. The sentences were recorded by four different speakers (Tom, Markus, Ork and Frans), resulting in a set of almost 600 sentences (some sentences were deleted in the segmentation process, because a word was skipped by the speaker).

Initially the speakers were to imitate the pronunciation of the NeXTeNS speech synthesizer for Dutch[6]. Speakers tend to change their pronunciation when they have to enumerate a large amount of similar sentences. This problem could have been avoided by letting the speakers repeat the speech synthesizer for each sentence. Also, by using Nextens's automatically generated phonetic transcription, the same phonetic transcription would have been available for every speakers' sentences. It would have ensured consistent realizations among speakers.

However, the automatically generated phonetic transcriptions are not entirely based on Nextens, since name transcriptions were constructed manually. Secondly, NeXTeNS is not installed on the computer in the recording studio at the Institute of Phonetic Sciences. Because of these two problems, NeXTeNS was not used for the recording session. This results in a more noisy training and test set, since the phonetic transcriptions do not exactly match the actual speech waveform. On the other hand, the recorded speech is more natural.

As the toolkit does not require phoneme duration information for the training sentences, the (differences in) timing in the pronunciation of the training sentences is not important. The toolkit learns to recognise the phonemes through fitting the phonetic transcriptions on the training set. Therefore, the phoneme duration information that is generated by Nextens for each utterance was not needed anyway, just the phonetic transcriptions. These transcriptions are used for all realisations of the same sentence, even though there might be variation between speakers relative to the transcription, for instance dropping or inserting phonemes. In order to avoid that, the phonetic transcription should have been done manually, but this would

[6]for more information, see the reference [Mar]

take to much time for this project.

As mentioned before, the sentences were recorded at the recording studio of the Institute of Phonetic Sciences. This gave the opportunity to use a high-quality microphone in a silent environment. The speakers were given a list with sentences which they had to read aloud. After about every 50 sentences they took a short break, and drank a glass of water.

The differences in pronunciation between speakers (and their consequences) can be categorised in two categories:

Phonetic change E.g., 'n'-deletion: some speakers said `/~z e v @ /`, others `/ z e v @ n /`,

Articulation variation E.g., some speakers had a rolling 'r', others not in, for example, 'Ork'

Phonetic change degrades the quality of the training set, since the same phonetic transcription was used for all speakers. The main problem caused by phonetic change is that the model cannot properly train on phonemes because the example set for a particular phoneme is flawed. It might even cause overtraining, when the model considers for instance the pronunciation of `/ @ sil /` as one of the allophones of `/ @ /` followed by `/ n /`. For our limited domain this need not make the results worse, because the word `/ z e v @ /` is then recognised as `/ z e v @ n /`, i.e. the word "zeven", instead of being recognised as "zeve", which is not a word and thus leads to failure of meaning extraction. Methodologically speaken this is clearly erroneous, though.

Articulation variation on the other hand is of course a problem for recognition but if there were no articulation variation the task of recognising would become an instance based learning problem (much more easier than it in fact is). So this is not a flaw of the data but an inherent aspect of the complexity of the field of speech recognition.

The recordings were segmented into 590 different wav-files by hand, using "Praat" [Boe04]. Therefore the data can be assumed not to contain any extra noise caused by variation in silence length etc.

3.4 Phonetic Transcription

HTK uses so-called Master Label Files to store information associated to speech. Which makes things a bit confusing is the fact that there are two things an MLF can contain: words and phonemes. In the tutorial the usages of various HTK tools are shown that can convert lists of sentences into lists of words and then lists of phonemes, the last two in an MLF. The phonetic MLF is made using the dictionary.

For the IFA corpus, annotation was already provided, so these were converted into a phonetic MLF straight away with a tool that Rob van Son provided and was adapted to the special syntax of HTK MLFs.

The generated test sentences were indeed turned into MLF's as described in the HTK Book. It turned out that the phonetic representations used were not completely the same. This had to be solved twice.

1. Because some of the transcriptions of the IFA corpus contained phonemes that were not in the dictionary, a discrepancy arose that caused problems in the subsequent steps. As HTK builds a phoneme list using the dictionary, it ran into errors assuming that the symbols in the transcriptions could not be phonemes. To solve this ad hoc, the missing phonemes `2, o+, X, Y` and `F` were added to the phoneme list generated by HTK.

2. The phonemes containing digits and/or a + sign could not be parsed directly from an MLF file. This is because the HTK toolset sees it as a number. This was fixed by replacing them.

4 Controlling Training and Testing

The speech recogniser we built was suspected to have flaws. We have therefore built in various levels of restriction in the training and testing sets. Evaluation was done on certain combinations.

4.1 Training

In principle, the HMM should be tested on a large corpus containing a wide range of phoneme pronunciations. For this, we used a 1000 sentences from the IPA corpus. These were annotated already. The speakers of this set were spread over gender and age, so the corpus can be seen as quite representative for all speakers.

A second training set was more domain-specific and also speaker-specific. The team members, except for Marijn and Daan, recorded 150 sentences from the grammar. As all team members are students that started in 1999, male and living in or near Amsterdam, this would give the HMM more information on that particular kind of speaker. This made it possible to take the interspeaker variation into account when testing, by testing on one speaker (that then has to be left out of the training set).

4.2 Testing

To test the HMM's, we made a few different pairs, were the first element is the way the HMM was trained and the second how it was tested. Note that it was not possible to train on just IFA or Domain and test on the other, as the phoneme sets were disjunct.

TRAINED ON	TESTED ON
IFA & Domain	Domain, training speakers
IFA & Domain	Domain, 'unknown' speaker
IFA & Domain	New sentences, training speakers
IFA & Domain	New sentences, new speaker

Table 1: The different training and testing methods

5 Results

As noted above, there were several seperations of training- and testset. In this section we will discuss the results of these different divisions.

The simplest test is leaving out a percentage of the recorded sentences while training. The recognition rates for different sizes of the test set are as follows:

	WORD	SENTENCE
PERCENTAGE	RECOGNITION (%)	RECOGNITION (%)
10	99.71	91.38
20	99.46	92.31
50	99.67	89.93
80	99.66	89.18

Table 2: Testing on random sentences

Another test was to train the model leaving all sentences of a particular speaker out completely. The test was then run on that speaker's sentences. This test gives an indication of how robust the system has generalised from speaker-specific phoneme pronunciations.

In the third performed test, some sentences were left out of the training set entirely. Thus, the test set consisted of sentences that were never seen before.

Also, a small test was performed in which all sentences of a particular speaker were removed from the training set, as well as a certain number

LEFT OUT SPEAKER	WORD RECOGNITION (%)	SENTENCE RECOGNITION (%)
Tom	99.57	85.71
Markus	99.78	72.60
Ork	99.43	89.13
Frans	99.78	81.63

Table 3: Testing on a new speaker

PERCENTAGE	WORD RECOGNITION (%)	SENTENCE RECOGNITION (%)
12	99.41	92.86
25	99.80	90.57
50	99.84	89.35

Table 4: Testing on new sentences

of sentences for all speakers (i.e. a combination of the previous two tests). Because there was not much variation between speakers, this was done only for one speaker (Tom). This produced the following result(s):

WORD RECOGNITION (%)	SENTENCE RECOGNITION (%)
99.57	84.35

Table 5: Testing on a new speaker, with new sentences

6 Discussion

6.1 Performance and stability

(...) [T]o build robust acoustic models, it is necessary to train them on a large set of sentences containing many words and preferably phonetically balanced. For these reasons, the training data will consist of English [in this case, Dutch] sentences unrelated to the phone recognition task. [You02]

The results suggest that the corpus is not only necessary, but also sufficient. Training the system almost exclusively on the IFA corpus does not affect the recognition rate. As a result of this, it did not matter much how the recorded data was divided over training and testing set.

Note that the language model enhanced performance as well, because of the very strong restrictions it imposed upon the language. Nevertheless, it does not give any indication what telephone number was to follow. Every new digit was a surprise to the system; this did not matter that much for performance. The performance on digits was facilitated by the fact there are only ten digits.

As shown, the system can handle speakers that it did not train on, but of course, all speakers were male students aged around 23, as mentioned before. The same holds for sentences[7] it did not train on.

6.2 Future extensions

Unfortunately, the scope of the project did not allow for testing with the language model disabled. This would have isolated the phoneme recogniser as such, and would probably also have meant for the training data on the domain to become more valuable for the system. This would be an interesting test to run.

Secondly, a less restrictive language model should be implemented. It can illustrate the interaction between HMM phoneme recognition and language model matching better; the present system depends heavily on the language model.

[7]In this case, 'a sentence' is: all realisations of one text.

A Recorded Sentences

These sentences were recorded by Tom, Frans, Ork and Markus at the Institute of Phonetic Sciences (University of Amsterdam) on the 18th of June 2004.

Due to a small bug in the grammar, a couple of sentences are in fact meaningless, but that should not bother the speech recogniser.

```
1. DRAAI DRIE ZEVEN NEGEN VIER ZES ACHT TWEE VIJF VIJF NEGEN
2. BEL EEN VIJF ZEVEN NEGEN
3. BEL VIER NUL NEGEN DRIE VIER
4. DRAAI ZEVEN VIER VIJF DRIE ZEVEN TWEE ACHT ACHT VIJF
5. DRAAI ZES NUL DRIE DRIE VIJF ACHT
6. BEL ORK DE ROOIJ
7. BEL TOMAS LENTZ
8. BEL FRANS ADRIAANS
9. BEL ROB VAN SON
10. DRAAI DRIE VIER VIJF EEN ZES VIER NEGEN VIER VIER
11. BEL DE ROOIJ
12. BEL DE ROOIJ
13. DRAAI ACHT VIER EEN DRIE NUL NUL NUL NUL
14. DRAAI VIER DRIE TWEE NUL ACHT NEGEN
15. BEL TOMAS LENTZ
16. BEL ZEVEN ZEVEN NEGEN NEGEN NEGEN ZES VIER ZEVEN
17. BEL VAN SON
18. BEL NUL TWEE NUL VIJF ZEVEN EEN VIER NEGEN VIJF NEGEN
19. BEL ORK DE ROOIJ
20. BEL DRIE VIER TWEE TWEE DRIE NUL ZES
21. DRAAI NUL ZES NEGEN ZES NUL NUL TWEE EEN ZES VIER
22. DRAAI TWEE VIER DRIE NEGEN
23. BEL NEGEN EEN VIJF ACHT DRIE ZES VIJF DRIE ACHT
24. BEL TWEE EEN VIJF NEGEN ZES DRIE ZES ZES
25. BEL FRANS ADRIAANS MARIJN KOOLEN
26. BEL VAN SON
27. DRAAI TWEE ZEVEN ZEVEN VIJF VIJF TWEE DRIE TWEE ACHT
28. BEL EEN NUL ZES TWEE NEGEN EEN DRIE TWEE VIJF ZEVEN
29. BEL DRIE TWEE VIJF NEGEN VIER VIJF NEGEN
30. BEL ZES NEGEN ACHT VIJF
31. BEL ORK DE ROOIJ
32. BEL FRANS KOOLEN
33. BEL TOMAS LENTZ
```

34. BEL EEN NUL EEN
35. BEL LENTZ
36. DRAAI ZEVEN ACHT ZES NEGEN EEN
37. BEL DAAN VREESWIJK
38. DRAAI VIJF EEN NUL DRIE ZES NUL NEGEN NEGEN ACHT
39. BEL DAAN
40. DRAAI DRIE NUL ZES EEN NEGEN DRIE VIJF ZES
41. DRAAI NUL EEN EEN EEN VIJF EEN VIJF EEN
42. DRAAI ZES ACHT NUL NEGEN VIER VIJF
43. BEL TWEE TWEE ACHT EEN
44. BEL LENTZ
45. BEL ZEVEN VIJF DRIE ACHT VIJF
46. DRAAI DRIE VIJF VIER ZEVEN ZES ZES EEN
47. BEL NEGEN TWEE ZEVEN NUL EEN EEN ACHT EEN TWEE ZEVEN
48. BEL TWEE ZES NUL
49. DRAAI DRIE NUL EEN VIER NUL
50. BEL VIER ACHT ZEVEN NUL ZEVEN DRIE VIJF
51. BEL VIJF ZES ZES DRIE
52. BEL NUL VIER DRIE ZES EEN ACHT EEN NEGEN ZES
53. BEL VIJF ZEVEN NEGEN NEGEN ZEVEN VIJF VIER VIER ACHT
54. BEL VIJF VIER VIER DRIE ZEVEN TWEE
55. BEL DRIE NUL ACHT ACHT NEGEN VIJF
56. DRAAI VIJF ZES NUL
57. BEL LENTZ
58. DRAAI ZEVEN TWEE ACHT NUL ZEVEN ZEVEN ZEVEN TWEE
59. DRAAI ZEVEN TWEE EEN NEGEN
60. BEL VIJF NEGEN NEGEN ACHT
61. DRAAI TWEE NUL TWEE VIJF ZEVEN TWEE
62. BEL NUL NUL NEGEN
63. BEL TWEE NUL ACHT NEGEN
64. BEL VIER NEGEN NUL TWEE DRIE EEN TWEE
65. BEL TWEE NEGEN EEN NEGEN VIJF VIJF VIJF NUL
66. BEL DAAN VREESWIJK
67. DRAAI ACHT ZEVEN NUL
68. DRAAI ZEVEN VIJF NEGEN TWEE ACHT ZEVEN NEGEN ZEVEN DRIE ZES
69. BEL DAAN VREESWIJK
70. BEL ZEVEN VIJF ZES ACHT VIJF EEN NUL
71. DRAAI NEGEN EEN VIJF EEN ACHT VIER
72. BEL VIJF ZEVEN TWEE VIJF EEN VIJF NUL ZEVEN
73. BEL VAN SON
74. DRAAI VIER TWEE VIJF ACHT

75. BEL VIER ACHT VIJF DRIE VIER VIER ZES ZES
76. BEL TOM LENTZ
77. DRAAI VIER VIER VIJF VIER VIJF ZEVEN DRIE
78. BEL NEGEN DRIE VIJF VIER
79. BEL NEGEN ZES DRIE NUL ZES DRIE NEGEN ZES DRIE VIJF
80. DRAAI EEN NUL TWEE
81. BEL DAAN VREESWIJK
82. DRAAI ZEVEN DRIE NEGEN ZEVEN NEGEN ZEVEN VIER VIER VIJF
83. BEL ZEVEN VIJF ACHT VIJF EEN TWEE
84. BEL ZES VIJF NEGEN ACHT TWEE TWEE TWEE ACHT VIJF TWEE
85. DRAAI ZEVEN VIJF VIER
86. BEL MARIJN
87. BEL ACHT VIJF NUL
88. BEL TOM
89. BEL ACHT ZES NEGEN EEN TWEE NEGEN VIJF TWEE NUL
90. BEL LENTZ
91. DRAAI VIER DRIE ACHT EEN ZEVEN EEN ACHT ZEVEN NUL VIER
92. BEL ROB
93. BEL ZES NUL NEGEN VIJF ZES EEN TWEE ZEVEN EEN ZEVEN
94. BEL EEN VIJF ACHT NUL VIER VIJF DRIE NEGEN ACHT TWEE
95. BEL ZES TWEE TWEE EEN TWEE DRIE VIJF VIJF
96. BEL NUL TWEE EEN VIER ZEVEN NEGEN
97. DRAAI ZEVEN EEN ZES DRIE EEN NEGEN
98. BEL ZEVEN VIER ZES NUL ZEVEN
99. BEL LENTZ
100. DRAAI VIJF NUL ACHT
101. DRAAI VIJF NEGEN DRIE NEGEN
102. DRAAI ZEVEN VIJF NEGEN EEN ZES TWEE TWEE NUL
103. BEL EEN VIER VIER VIJF VIER EEN TWEE
104. BEL TOMAS LENTZ
105. DRAAI ACHT DRIE ACHT VIER EEN NUL NEGEN
106. BEL ROB VAN SON
107. BEL FRANS ADRIAANS
108. DRAAI NEGEN ACHT DRIE ZEVEN
109. BEL ZEVEN EEN ZEVEN ZEVEN ACHT EEN NUL ZES ACHT VIER
110. DRAAI EEN TWEE NEGEN ZES ACHT VIJF
111. BEL ROB VAN SON
112. BEL VAN SON
113. BEL DE ROOIJ
114. BEL DRIE NUL NUL ZEVEN NUL NEGEN ACHT
115. DRAAI NEGEN NUL ZES ZEVEN DRIE DRIE TWEE VIER ZEVEN ZEVEN

116. DRAAI NEGEN VIER VIJF ACHT VIER ACHT ZEVEN
117. BEL LENTZ
118. DRAAI EEN NEGEN VIJF EEN DRIE ZEVEN ACHT VIER
119. BEL ZES ZEVEN TWEE NEGEN EEN VIER EEN
120. DRAAI TWEE NEGEN ACHT DRIE
121. DRAAI NEGEN ZES NEGEN DRIE
122. DRAAI ACHT DRIE ZES
123. BEL ORK DE ROOIJ
124. DRAAI VIER VIER VIER
125. DRAAI NEGEN ZES DRIE NEGEN EEN TWEE ZES DRIE DRIE
126. DRAAI VIJF NUL ZEVEN EEN NUL NUL DRIE ACHT
127. BEL ACHT ZES VIJF ACHT
128. DRAAI TWEE EEN ZES TWEE EEN ZEVEN DRIE
129. DRAAI ZES EEN DRIE NEGEN NEGEN ZEVEN ZEVEN
130. BEL ZES NUL ZEVEN VIER NEGEN
131. DRAAI ZEVEN DRIE TWEE VIJF
132. BEL TOMAS LENTZ
133. BEL ACHT EEN TWEE DRIE ACHT ZEVEN ZES TWEE NEGEN
134. BEL VIJF TWEE NUL VIER NEGEN VIJF
135. BEL DRIE NEGEN NUL ZEVEN NEGEN VIER
136. DRAAI VIER ZES DRIE
137. DRAAI EEN ZEVEN NEGEN DRIE NEGEN VIER VIER VIER NEGEN
138. BEL VIER TWEE ZES ACHT
139. DRAAI VIER EEN TWEE ACHT EEN VIJF VIJF ZEVEN DRIE VIER
140. BEL EEN EEN ZES DRIE ACHT NEGEN VIER NEGEN
141. BEL ACHT VIER ACHT ZEVEN
142. BEL FRANS ADRIAANS KOOLEN
143. DRAAI DRIE ZES VIJF NEGEN VIER EEN ACHT EEN ZES EEN
144. DRAAI NUL NUL DRIE
145. DRAAI DRIE ZES NEGEN TWEE NEGEN TWEE VIER NUL
146. DRAAI VIER TWEE ZES ACHT DRIE ZEVEN
147. BEL NUL EEN DRIE
148. DRAAI VIJF NUL VIJF VIJF NUL ACHT
149. DRAAI VIJF ACHT ZEVEN
150. BEL VREESWIJK

B The scripts

Below is a list of all the scripts created for this project, with a short description. The code of the scripts should be informative enough to understand how they perform there task (for anyone who is interested).

`compare.py` Comparing test output with original sentences. It parses the recout.mlf file and a file with the original sentences, and outputs for every sentence whether it was recognised correctly or what went wrong.

`dividetesttrain.py` Dividing filelist into train- and testset. Either a percentage or the name of a speaker can be given as a command line argument to this file, creating a test set from a certain size or a certain speaker. The other command line arguments are the file of origin, the file for the train set and the file for the testset.

`ExtractPhonemesTomAndOrk.pl` Creating MLF's from IFA corpus. The transcription for the IFA corpus is turned into a Master Label File with phonemes. Uses `SentenceLabel.pl`.

`fixFonOrk.py` Rewriting phonemes. This scripts detects and rewrites illegal phonemes into legal ones. It can be run on dictionaries, phoneme lists and master label files.

`generateTestMLF.py` Adapting an MLF file for multiple speakers. This scripts reads the MLF file generated from the testprompts in step 4 and duplicates each label for each speaker that has recorded this label. The scripts outputs a new MLF containing the duplicated result.

`hmm4gen.py` Creation of the `silst` silence state for step 7

`hmmdefsgen.py` Conversion from `proto` into an `hmmdefs` (see [?] for more details on the formats). The scripts automatically copies the prototype model in *hmm0* for every monophone and creates a `hmmdefs` file. Strangely enough, there was no such script in the HTK toolset.

`init_stp` Sets the HTK environment variables. Please check the variables in this script, and change them for your HTK installation.

`maketrihed` Creating mktri.hed. Used to create a .hed file for step 9.2.

`quotePhone.py` This script quotes triphones, i.e. puts quotes around them. It is useful to quote out triphones which contain illegal characters (HTK does not understand numbers and cannot handle the +-sign very well).

Nevertheless, this script seems insufficient to solve all HTK format problems. In the current implementation; it is no longer used because all the illegal phonemes are rewritten to legal ones before starting with `fixFon.py`.

`SentenceLabel.pl` Needed for `ExtractPhonemesTomAndOrk.pl`.

`transdict.py` Dictionary translation script, that translates dictionaries from other formats (IPA, Nextens) into the HTK format. It sorts all the entries and converts words to upper case, which seems to be preferred by HTK. N.B.: this script can also be used to substract two dictionaries from each other.

References

[Boe04] Boersma, P., D. Weenink. Praat. Internet: http://www.praat.org, 2004.

[Cam02] Cambridge University Engineering Department, Microsoft Corporation. Hidden Markov Model Toolkit. Internet: http://htk.eng.cam.ac.uk, december 2002.

[Cha95] Chan, D., A. Fourcin, D. Gibbon, B. Granstrom, M. Huckvale, G. Kokkinakis, K. Kvale, L. Lamel, B. Lindberg, A. Moreno, J. Mouropoulos, F. Senia, I. Trancoso, C. Veld, J. Zeiliger. "EU-ROM - A Spoken Language Resource for the EU". In *Eurospeech '95*, volume 1, pages 867–870. 4th European Conference on Speech Communication and Speech Technology, September 1995.

[Mar] Marsi, E., A. van den Bosch, J. Kerkhoff, T. Rietveld, A. Russel, E. Klabbers. NeXTeNS: Open Source Text-to-Speech for Dutch. Internet: http://nextens.uvt.nl.

[You02] Young, S., G. Evermann, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, P. Woodland. The HTK Book. Internet: http://htk.eng.cam.ac.uk, december 2002.