

# 050/520/600.666 Information Extraction from Speech and Text

Project # 2

Due March 17, 2006.

The goal of this project is to build a small-vocabulary isolated-word recognizer. You will create *fenonic baseforms* for each word in a 53 word vocabulary, as described in Chapter 3, estimate the hidden Markov model (HMM) parameters, and evaluate the recognition performance of the resulting HMM system.

## Training and Test Data Files:

There is a file enumerating the possible values of the *fenemes*, the (quantized) outputs of the acoustic processor.

`jhucsp.lblnames` contains 306 five-character-long feneme names, one per line, resulting in 306 lines, plus a title-line at the top of the file. [Total: 307 lines]

There are three training-data files as described below.

`jhucsp.trnscr` is the “script” that was read by the speakers whose speech comprises the training data. Each of the 53 words in the vocabulary were presented, in some randomized order, to speakers, and there were 10 such speakers, so that the script-file contains 10 blocks of 53 words, for a total of 530 lines of data, plus a title-line at the top of the file. [Total: 531 lines.]

`jhucsp.trnlbls` contains the “labels” or fenemes, one (long) feneme-string per line, corresponding to the utterance of each word in the script file described above. There are 530 lines, plus a title-line at the top of this file as well. [Total: 531 lines, 132,083 fenemes.]

`jhucsp.endpts` contains “end-point” information, or the information about the leading- and trailing-silence surrounding each utterance. This information is encoded in the form of two integers per line, say,  $i$  and  $j$ , to indicate that the *last feneme of the leading silence* is at position  $i$ , the *first feneme of the trailing silence* is at position  $j$ , and the speech corresponds to the  $(i + 1)$ -th through  $(j - 1)$ -th labels in the label-file. There are, again, 530 lines of data, plus a title-line. [Total: 531 lines.]

Finally, there is a test-data file,

`jhucsp.tstlbls` that contains the labels', one variable-length *feneme* label-string per line, corresponding to an utterance of each word in the test set. There are 530 lines, plus one title-line at the top of this file as well. [Total: 531 lines, 185,396 fenemes.]

Proceed to build the recognizer as follows.

1. **Create elementary (fenonic) models:** For each of the 306 possible values of the fenemes, create an elementary HMM of Figure 3.4 in Chapter 3.

Initialize these 2-state HMMs or *fenones* by making the feneme corresponding to the fenone have output probability 0.5 on each output producing arc, and by making the probability of a 1-step transition through the HMM have probability 0.8; make the remaining outputs and transitions equiprobable. Specifically, in Figure 3.4, set

$$p(t_1) = 0.8, \text{ and } p(t_2) = p(t_3) = 0.1, \quad (1)$$

and

$$q(\text{feneme}|t_1) = q(\text{feneme}|t_2) = \begin{cases} 0.5 & \text{if feneme} = \text{fenone} \\ \frac{0.5}{305} & \text{if feneme} \neq \text{fenone} \end{cases} \quad (2)$$

These probabilities will, of course, be re-estimated from the training data.

2. **Create a silence model:** To model the leading and trailing silence around words, create a special fenonic HMM named `<sil>` with the 7-state topology of Figure 3.1.

Initialize the silence HMM by making all transition leaving a state equally likely and, on each output producing arc, making all the 306 fenemes equally likely.

3. **Pick fenonic baseforms:** Use the fenemic-string from the non-silence portion of the *first* instance of each word as the *singleton fenonic baseform* (cf Section 3.6).
4. **Form word HMMs:** Construct an HMM for each of the 53 words by concatenating the elementary (fenonic) HMMs of Step 1 based on its fenonic baseform of Step 3. Append the `<sil>` HMM to both the beginning and the end of this composite HMM to create the word HMM.

Note that even though you used two “copies” of the `<sil>` HMM, and may have used more than one copy of the same fenonic HMM in composing the word HMM, these should all be considered to have *tied* parameters. They are also tied across their use in word HMMs of all 53 words. In other words, when collecting the counts (cf equations (27) and (28) in Chapter 2), there should be only as many accumulators as there are free parameters in Steps 1 and 2 above.

5. **Train the word HMMs:** Use the training data files to train all the word HMMs. In particular, ignore the end-point information at this stage, and train the silence model together with the fenonic models. (The end-point information was only used for obtaining reasonable singleton baseforms in Step 3.)

The training data consists of 10 tokens of each of the 53 words, which may be treated as independent realizations. Therefore, unlike Project # 1, you will run the forward-backward algorithm separately on each of the 530 utterances. But a common set of counters  $c^*(t)$  and  $c^*(y|t)$  will accumulate the posterior probabilities  $P(t^i = t)$ , and you will carry out the parameter updates (cf equations (36) and (37), Chapter 2) only after you have completed all 530 forward-backward passes. More formally

- (a) Number the fenonic models lexicographically from 1 to 306, and let the `<sil>` model be numbered 307. Let the  $j$ -th arc in the  $i$ -th fenonic model be denoted  $t_{ij}$ . Note that  $j = 1, 2, 3$  when  $i = 1, \dots, 306$  (cf Figure 3.4), and  $j = 1, \dots, 12$  when  $i = 307$  (cf Figure 3.1).

For each arc  $t_{ij}$ , establish a counter  $c_{ij}$  and a 306-sized array of counters  $c_{ij}[a]$ , where  $a$  runs over the fenemic alphabet in `jhucsp.lblnames`, and initialize them to be zero. (If  $t_{ij}$  is a null transition,  $c_{ij}[a]$  will remain zero throughout the re-estimation process.)

- (b) Jointly sort the lines in the files `jhucsp.trnscr` and `jhucsp.trnlbls`, so that all 10 utterances of a word follow each other. This way, you can construct the trellis for a word HMM once, and use it 10 times while processing the 530 training utterances, one forward-backward pass per utterance. A training utterance increments the counters  $c_{ij}$  and  $c_{ij}[\cdot]$  of all arcs  $t_{ij}$  used in the HMM of that word.
- (c) After all 530 forward-backward passes are completed, update the HMM parameters of the fenonic models as

$$p(t_{ij}) = \frac{c_{ij}}{c_{i1} + c_{i2} + c_{i3}}, \quad j = 1, 2, 3, \quad \text{and} \quad p(a|t_{ij}) = \frac{c_{ij}[a]}{c_{ij}}, \quad j = 1, 2, \quad (3)$$

for  $i = 1, \dots, 306$ . The parameters of the `sil` model are updated similarly.

6. **Observe convergence:** Plot the log-likelihood of the 530 utterances as a function of the number of iterations of the parameter update (3). Check when the likelihood stops increasing. (4-8 iterations should suffice.)
7. **Test the HMM system:** For each of the 530 utterances in the test data-file, compute the forward-probability of the acoustics (feneme-string) under each of the 53 word models, and pick the word with the highest likelihood.

Submit the following as your report for this project.

1. A plot of the training data log-likelihood as a function of the number of iterations from Step 6 above.
  - It is customary in speech recognition to report the average per-frame log-likelihood, which is the total log-likelihood divided by the number of acoustic observations in the training data.
2. For each utterance in the test data, the identity of the most likely word and a *confidence*.

- To calculate the confidence value, divide the forward-probability of the most likely word by the *sum* of the forward probabilities of all the 53 words (including the most likely word).
3. Your source code, along with substantial documentation about exactly what files (among the 5 data files provided for the project) are needed to run each module, and the command line (usage) for running the training and testing modules.
    - Your code should expect the 5 files to be in the current directory, and should preferably work on a **i686** machine running a recent version of Redhat linux.

A *tarball* containing all the above, with obvious filenames and a README are expected.