(520|600).666
Information Extraction from Speech and Text

Project # 2

Due March 23, 2015.

The goal of this project is to build a small-vocabulary isolated-word recognizer. You will create *fenonic baseforms* for each word, as described in Chapter 3, of a 48 word vocabulary, estimate their 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.

`clsp.lblnames` contains 256 two-character-long feneme names, one per line, resulting in 256 lines, plus a title-line at the top of the file. [Total: 257 lines]

There are three training-data files as described below.

`clsp.trnscr` is the "script" that was read by the speakers whose speech comprises the training data. Each of the 48 words in the vocabulary were presented to speakers in some randomized order. The script-file contains at least 10 and up to 25 examples of each word, for a total of 798 lines of data, plus a title-line at the top of the file. [Total: 799 lines.]

`clsp.trnwav` contains the name of the speech (waveform) file corresponding to the utterance of each word in the script file described above. There are 798 lines, plus a title-line at the top of this file as well. [Total: 799 lines.]

`clsp.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 798 lines, plus a title-line at the top of this file as well. [Total: 799 lines, 106,785 fenemes.]

`clsp.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, 798 lines of data, plus a title-line. [Total: 799 lines.]

Finally, there is are two test-data files:

`clsp.trnwav` contains the name of the speech (waveform) file corresponding to the utterance of each word in the test set. There are 393 lines, plus a title-line at the top of this file as well. [Total: 394 lines.]

`clsp.devlbls` contains the labels, one variable-length *feneme* label-string per line, corresponding to an utterance of each word in the test set. There are 393 lines, plus one title-line at the top of this file as well. [Total: 394 lines, 52,812 fenemes.]

Proceed to build your *primary* recognizer as follows.

1. **Create elementary (fenonic) models**: For each of the 256 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 setting the 1-step, output producing transition through the HMM to have probability 0.8; set the remaining outputs and transitions to be equiprobable. Specifically, in Figure 3.4, set

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

   and for a fenone whose HMM-name is `y`,

   $$q(y|t_1) = q(y|t_2) = \begin{cases} 0.5 & \text{if } y = \texttt{y} \\ \frac{0.5}{255} & \text{if } y \neq \texttt{y}. \end{cases} \tag{2}$$

   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 256 fenemes equally likely.

3. **Pick fenonic baseforms**: Use the fenemic-string from the non-silence segment 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 48 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 48 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 the non-silence segment for the singleton baseforms in Step 3.)

   The training data consists of 10-25 tokens of each of the 48 words, which may be treated as independent realizations. Therefore, unlike Project # 1, you will run the forward-backward algorithm separately on each of the 798 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 798 forward-backward passes. Specifically,

   (a) Number the fenonic models lexicographically from 1 to 256, and let the `<sil>` model be numbered 257. Let the j-th arc in the i-th fenonic model be denoted t_{ij}. Note that $j = 1, 2, 3$ for $i = 1, \ldots, 256$ (cf Figure 3.4), and $j = 1, \ldots, 12$ for $i = 257$ (cf Figure 3.1).

   For each arc t_{ij}, establish a counter c_{ij} and a 256-sized array of counters $c_{ij}[y]$, where y runs over the fenemic alphabet in `clsp.lblnames`, and initialize them to be zero. (If t_{ij} is a null transition, $c_{ij}[y]$ will remain zero throughout the re-estimation process.)

   (b) Jointly sort the lines in the files `clsp.trnscr` and `clsp.trnlbls`, so that all 10-25 utterances of a word follow each other. This way, you can construct the trellis for a word HMM once, and use it 10-25 times while processing the 798 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 798 forward-backward passes are completed, update the HMM parameters of the fenonic models as

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

   for $i = 1, \ldots, 256$. The parameters of the `<sil>` model are updated similarly.

6. **Observe convergence**: Plot the log-likelihood of the 798 utterances in the training data-file as a function of the number of iterations of the parameter update (3). Check when the likelihood stops increasing. (Do 4-8 iterations suffice, or do you need 100s of iterations?)

7. **Test the HMM system**: For each of the 393 utterances in the test data-file, compute the forward-probability of the acoustics (feneme-string) under each of the 48 word models, and pick the word with the highest likelihood.

Submit the following in your report for the *primary* system for this project.

⇒ 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.

⇒ 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 48 words (including the most likely word).

⇒ 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 run on a `x86_64` machine running a recent version of GNU/Linux.

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

Build a *contrastive* system to explore alternatives to your primary system. Some suggestions are provided below, but you are free to innovate for this part of the project.

(i) Instead of using all 798 training utterances in Step 5 and relying on the likelihood convergence of Step 6 to decide when to stop HMM training, you may want to base the number of iterations of HMM training on recognition accuracy.

Randomly divide the training data into an 80% *kept* portion and a 20% *held-out* portion, taking care to balance the distribution of word-tokens between the two: i.e. aim to get 20% of the instances of each word into the held-out set, retaining 80% in the kept set.

Build the system according to Steps 1-5, but using only the kept data. After each iteration of Step 5, test the resulting system <u>on the held-out data</u> as described in Step 7, and measure accuracy. Stop HMM training when the accuracy stops increasing and note how many iterations were needed. Let this be N^*.

Now combine the kept and held-out data and repeat Steps 1-5 and 7, using N^* iterations of HMM training in Step 5 regardless of what happens to the log-likelihood in Step 6.

(ii) After building your primary system, you may want to revisit Step 3 and instead of picking as the baseform for each word the fenemic-string of the *first* instance of that word, choose among all 10 instances of the word, as described in Equation (2) of Section 3.8 in the book. Redo Steps 1-2 and 4-7, choosing this alternative baseform in Step 3.

(iii) Instead of building fenonic baseforms, you may skip Step 3 and simply use the *spelling* of a word as its baseform (!), with one elementary HMM for each letter of English.

Since Step 1 is invalid if the elementary HMMs are not associated with fenemes, you will have to initialize the elementary HMMs differently. You may take inspiration from the initialization used in Project #1.

Redo Steps 2 and 4-7 with these spelling-driven baseforms in Step 3. Compare the log-likelihood plot of Step 6 with what you saw in Project #1.

Submit an analogous report for your *contrastive* system (see items marked $\Rightarrow$ above).