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

Project # 2

Due April 11, 2024.

The goal of this project is to build an isolated-word recognizer for vocabulary of 48 words.

- The speech signal has been processed to extract spectral features from small (25ms) windows at a fixed frame-rate (100 frames/sec), and the resulting feature-vectors have been quantized into to 256 clusters, similar to what you did in Homework # 1. These will constitute your discrete-valued observation sequences $\mathbf{Y}$.
- Define hidden Markov models (HMMs) for individual letters, and an HMM for silence.
- Compose the HMM for each word by concatenating the HMMs of its spelling—for example, `also` ← `[a] [l] [s] [o]` and `into` ← `[i] [n] [t] [o]`.
- Use the training samples to estimate the parameters of the letter and silence HMMs.
- Recognize the test samples using the resulting set of HMMs.
- Evaluate the system's recognition accuracy.

Training and Test Data Files:

There is a file enumerating the possible values of the 256 *cluster labels*, i.e. the (quantized) outputs of the acoustic processor.

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

There are four training-data files and two test 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 $\mathbf{Y}$, the processed speech corresponding to the utterance of each word in the script file described above. There is one long label-string per line, and there are 798 lines, plus a title-line at the top of this file as well. [Total: 799 lines, 106,785 labels.]

`clsp.endpts` contains "end-point" information, i.e. information about the leading- and trailing-silence surrounding each spoken word. This information is encoded in the form of two integers per line, say, i and j, to indicate that the *last label of the leading silence* is at position i, the *first label 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.]

`clsp.devwav` 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 $\mathbf{Y}$, one variable-length 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 labels.]

Only `clsp.trnscr`, `clsp.trnlbls` and `clsp.endpts` are used for HMM training; and only `clsp.devlbls` for testing. The speech (`*.wav`) files listed in `clsp.trnwav` and `clsp.devwav` are provided for your listening pleasure.

Proceed to build your *primary* recognizer as follows.

1. **Create the HMM for each letter**: The words in this vocabulary do not use `[k]`, `[q]` and `[z]`. For the remaining 23 letters, create a 3-state HMM, with a left-to-right topology corresponding to the transition probability matrix

$$\begin{bmatrix} 0.8 & 0.2 & 0.0 \\ 0.0 & 0.8 & 0.2 \\ 0.0 & 0.0 & 0.8 \end{bmatrix},$$

   where an emission distribution is attached to each of these (5) arcs, and where the remaining (0.2) transition probability on the last state corresponds to a non-emitting (null) arc for *exiting* this HMM.

   Note that each emission distribution is on an output alphabet of size 256. Initialize all the emission distributions using the "unigram" frequency of the training labels in `clsp.trnlbls`. (Does this ensure that no label has zero probability?)

2. **Create an HMM for silence and non-speech sounds**: Create a 5-state HMM named `SIL` for modeling the silence (or noise) surrounding the words, with a topology corresponding to the transition probability matrix

$$\begin{bmatrix} 0.25 & 0.25 & 0.25 & 0.25 & 0.00 \\ 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.00 & 0.25 & 0.25 & 0.25 & 0.25 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.75 \end{bmatrix},$$

where an emission distribution is attached to each of these (17) arcs, and where the remaining (0.25) transition probability on the last state corresponds to a non-emitting (null) arc for *exiting* this HMM.

Initialize all the emission distributions using the unigram frequency of the training labels from the leading- and trailing-silences in `clsp.trnlbls`, as indicated by `clsp.endpts`. If needed, add 1 to each unigram count to make sure that no label gets zero probability.

3. **Create grephemic baseforms**: Use the spelling of each word as its "pronunciation," i.e. as the sequence of sub-word units whose concatenation generates the acoustics of the word.

4. **Form word HMMs**: Construct an HMM for each of the 48 words by concatenating the letter HMMs of Step 1 based on its baseform in Step 3. Append the `SIL` HMM of Step 2 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 letter 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 per equations (27) and (28) in Chapter 2, there should be only as many accumulators as there are transition and emission probabilities 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 from this stage onwards, and train the silence model together with the letter models. (The end-point information was only used for initializing the silence HMM in Step 2.)

   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 of equations (36) and (37) in Chapter 2 only after you have completed all 798 forward-backward passes. Specifically,

   (a) Number the letter HMMs lexicographically from 1 to 23, and let the `SIL` HMM be numbered 24.

   Let the j-th arc in the i-th HMM be denoted t_{ij}. Note that $j = 1, 2, \ldots, 6$ for $i = 1, \ldots, 23$, and $j = 1, 2, \ldots, 18$ for $i = 24$.

   For each arc t_{ij}, establish a counter c_{ij}, and initialize it to zero.

   For each non-null arc t_{ij}, establish a 256-sized array of counters $c_{ij}[y]$, where y runs over the labels, and initialize them to zero. No $c_{ij}[y]$ are needed for null arcs.

(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 for successive training utterances of a word, 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 parameters of the 23 letter HMMs as

$$p(t_{ij}) = \frac{c_{ij}}{\sum_{j' \,:\, L(t_{ij'})=L(t_{ij})} c_{ij'}} \qquad \text{and} \qquad p(y|t_{ij}) = \frac{c_{ij}[y]}{c_{ij}}, \qquad j = 1, \ldots, 5, \quad (1)$$

for $i = 1, \ldots, 23$.

Similarly update the transition probability $p(t_{i6})$ for all the null arcs of the letter HMMs; no emission probability $p(y|t_{i6})$ is associated with them.

Update the parameters of the `SIL` model similarly.

6. **Observe convergence**: Plot the total log-likelihood of the 798 training utterances as a function of the number of iterations of the parameter update (1). Check when the likelihood stops increasing. (Do 10s of iterations suffice, or do you need 100s of iterations, as you probably did in Project #1?)

7. **Test the HMM system**: For each of the 393 utterances in the test data-file, compute the forward-probability of the acoustics (label 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 7 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 7 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 on the held-out data 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) Instead of using graphemic baseforms in Step 3, you may use a hand-crafted pronunciation dictionary (e.g. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`) , and create HMMs for each phoneme of English.

Redo Steps 1-7 with these phonemic baseforms in Step 3. Compare the log-likelihood plot of Step 6 with the corresponding plot for graphemic baseforms.

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