

**050.680 — Learning Theory**  
**Problem Set 5**  
**Due 27 April 2007**

In this problem set, you will explore the use of a Markov chain as a model of English phonotactics. We have provided to you a Matlab function `phone_bigram` which takes as input a file containing phonetically transcribed text and returns a vector of three arguments: a list of the phones occurring in the file, a vector containing the counts of the individual phones and a matrix containing the counts for each phone bigram (where position  $(i, j)$  contains the number of occurrences of the bigram  $w_i w_j$ , and  $w_i$  is the  $i$ th member of the phone list). You also have available to you the following files of phonetically transcribed data to use for training:

- `aesop.trans` is a transcription of Aesop's fables.
- `aesop-short.trans` is a transcription of the first 1096 words of Aesop's fables.
- `lexicon.trans` is a lexicon of approximately 60K English words (the CMU Pronouncing Dictionary, available at <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>)
- `KF-1002.trans` is a lexicon of the 1002 most frequent English words (according to Kučera and Francis's (1967) counts)

These texts have all been phonetically transcribed using the Arpabet, a set of two letter sequences that represent the phonemes of English. For information about the Arpabet, see <http://www.stanford.edu/class/linguist238/fig04.01.pdf> for the consonants and <http://www.stanford.edu/class/linguist238/fig04.02.pdf> for the vowels.

**Part A:** Use the `phone_bigram` function to compute the bigram counts for each of these files. Now, write a function `MLE_prob` that takes such a bigram count matrix as input and computes from it a maximum likelihood estimate of the probability distribution for the bigram.

**Part B:** Also available to you are two files of pseudo words (called `good_words` and `bad_words`) that are drawn from a paper by Frisch, Large and Pisoni (2000), who found differences in speakers willingness to judge these two sets words as possible English words. Using the function `word_prob_batch`, which takes a filename, list of phones, and a bigram probability distribution, and returns a vector of probabilities for each word in the file, compute the probabilities for these files. Does your bigram model distinguish all of the good words from all of the bad ones? How about on average? Do different training sets behave differently? That is, is there any difference if we train on large or small training sets? What differences do you observe if we use a lexicon of unique words as opposed to a text whose bigram distributions are affected by word frequency?

**Part C:** You will likely have noticed that your model sometimes assigns zero probability to phone sequences, as there are certain bigrams that fail to occur in the training data. To avoid this, implement a function `add1_prob` that works like your previous function `MLE_prob`, but which uses add-1 smoothing. How do your answers from part B change if you use these new probability estimates?

**Part D:** By using the probability that is assigned by the bigram model to a string as a measure of the goodness of that string, we are putting longer strings at an inherent disadvantage. Another measure of string goodness that we can use instead is the average bigram probability over a string. Write a function `avg_prob_batch` which computes average bigram probability for each word in a file rather than the product of the bigram probabilities. Does this change affect your results?

**Part E: (Optional)** One way of assessing the goodness of a statistical model is by computing its perplexity over the data. Perplexity for a bigram model is defined as

$$PP(\text{corpus}) = 2^{-\frac{1}{Q} \sum_{i=1}^Q \log P(w_i | w_{i-1})}$$

where  $Q$  is the number of “words” of test data (where “words” is understood as phones here). Write a function which computes perplexity given a file of data, a list of words, and a bigram probability distribution. You may find it helpful to use `word_prob_batch` for some guidance. How do the models trained on the different data sets fare in terms of perplexity?

**Part F: (Optional)** You may recall that Hayes and Wilson in their paper had their hand at modeling another data set of word “grammaticality” judgments, from Scholes (1968). The pseudo words that Scholes used, along with the number of experimental subjects who judged each word as grammatical are available in the file `scholes-data.txt`. You can load this data into Matlab by using the following command:

```
[wordlist ratings] = textread('scholes-data.txt', '%s %d', 'delimiter', '\t')
```

Explore the goodness of fit between the different bigram-based models you have constructed and the ratings Scholes obtained.

**Part G: (Optional)** Modify the `phone_bigram` function so that it computes counts of phone trigrams. Compare the resulting trigram model to the bigram model trained on one or more of the datasets (with and without smoothing), with respect to predictions concerning possible words and perplexity.