

LING115 Homework #8
Due October 22, 2010

Instructions:

Create a `hw8` directory under your home directory. Make sure you save the programs you wrote for question 1 below in that directory. Save the write-up of your answers for question 2 in a file named `hw8.q2` under the `hw8` directory.

1. [3 pts] Padding

The biphone frequency model described in the lecture note does not care about which phoneme begins a word and which phoneme ends a word. According to that model, the biphones in `D AO G` 'dog', for examples, would be `D-AO` and `AO-G`. The model does not mention that the word begins with a `D` and ends with a `G`. But such information is necessary to explain why `[ŋap]` or `[sih]` cannot be possible word in English. No word in English begins with `[ŋ]`. No word in English ends with `[h]`.

One way to include such information in the biphone model is to pad a given phonetic transcription with word-boundary symbols and average the frequencies of biphones in the padded form. For example, we would first pad `D AO G` as `# D AO G #` and average the frequencies of four biphones: `#-D`, `D-AO`, `AO-G`, and `G-#`.

Copy the two programs `count_biphones.py` and `wordlikeness.py` under `/home/ling115/python_examples` into your `hw8` directory. Modify the two programs so that the programs work with the biphones of padded phonetic transcriptions. In addition to the biphone frequencies `count_biphones.py` already keeps track of, your new version of the program should also keep track of the frequencies of biphones that begin with `#` (e.g. `#-D` in `# D AO G`) and those that end with `#` (e.g. `G-#` in `D AO G`). Similarly, your new version of `wordlikeness.py` should include the frequencies of such biphones at the edge (e.g. `#-D` and `G-#`) in calculating the average of biphone frequencies.

2. [3 pts] Correlation

How good is our model in predicting the degree to which a new word sounds like an English word? Ideally, the model's prediction should correlate well with the judgment made by native speakers of English. One could run an experiment with a bunch of new words where the native speakers rated whether each word sounds like an English word on some scale (e.g. 5-point scale, 0 for 'does not sound like an English word at all' and 5 for 'totally sounds like an English word'). The model should return high average frequencies for words that were rated high and low average frequencies for those that were rated

low. In sum, the correlation between the average biphone frequencies and the ratings by the native speakers should be high.

How do we calculate the correlation between the two with Python? One way is to use `RPy`, which is a Python interface to a widely used statistical programming language called `R`. Skipping the details about `R` and `RPy`, here's how you can calculate the correlation between the values of two different variables (e.g. speaker's rating vs. biphone frequency), say `[2,3,4]` and `[200,300,400]`.

```
>>> import rpy
>>> x = [2,3,4]
>>> y = [200,300,400]
>>> rpy.r.cor(x,y)
1.0
```

The `rpy.r.cor` function returns the correlation coefficient, a measure of the correlation between the values of two variables. If the two variables are positively correlated - positive in the sense that a high value for one variable corresponds to a high value for the other variable - the coefficient is between 0 and 1, with a higher coefficient suggesting a stronger correlation between the two. If the two variables are negatively correlated - negative in the sense that a high value for one variable corresponds to a low value for the other variable - the coefficient is between -1 and 0, with a coefficient closer to -1 suggesting a stronger negative correlation between the two.

Anyway, `/home/ling115/hw8_out/nonsense_ratings` have a set of nonsense words along with their average ratings collected from native speakers of English (adapted from Albright and Hayes, 2003). Let's call this nonsense-ratings from now. Each line in that file consists of (1) a phonetic transcription whose phonemes are separated by a single blank space and (2) its rating, separated from the phonetic transcription by a single tab space.

2.1. Use `count_biphones.py` and `wordlikeness.py` under `/home/ling115/python_examples/` to calculate the average biphone frequency for each of the words in the nonsense-ratings file. You'll have to improvise since the nonsense-ratings file looks slightly different from the input that `wordlikeness.py` expects. What is the correlation between the average biphone frequency of a word and its rating? Report the correlation coefficient using the `rpy.r.cor` function mentioned above.

2.2. Do the same as 2.1, except use the programs you wrote for question 1 - the ones where words are padded. What is the correlation between the two this time?