Assignment 2

L245

Text & Speech Encoding / Writers's Aids

Due Monday, February 6

1. [**from HW1:**] Here are some bigram probabilities (Figure 6.7, Jurafsky & Martin (2000), 1st edition). For example, $P(want|i) = 0.22$, whereas $P(i|want) = .0014$. Ignoring start & end probabilities, calculate the probabilities for the sentences (a) & (b) using a bigram model (show your work). Then, answer (c).

   (a) *i want to eat chinese lunch*

   (b) *i want to eat food*

		2nd word						
		i	want	to	eat	chinese	food	lunch
1st word	i	.0018	.22	.0020	.0028	.00020	.00020	.00020
	want	.0014	.00035	.28	.00035	.0025	.0032	.0025
	to	.00082	.00021	.0023	.18	.00082	.00021	.0027
	eat	.00039	.00039	.0012	.00039	.0078	.0012	.021
	chinese	.0016	.00055	.00055	.00055	.00055	.066	.0011
	food	.0064	.00032	.0058	.00032	.00032	.00032	.00032
	lunch	.0024	.00048	.00048	.00048	.00048	.00096	.00048

   (c) The sentence *i want to eat* is more likely than *i want to eat lunch*, yet both are good sentences. If we wanted a better grasp on the likelihood of a sentence actually appearing in the English language, what other properties might we need to account for in our model?

2. [**from HW1:**] Do question #7 from chapter 1 of the textbook (p. 30) ... with a few alterations/clarifications:

   - You need to ask 5 (or more) friends (or however many you need to sufficiently answer part b). You are still working with at least 10 bigrams.
   - Be sure to present your data (in a readable, organized format): you will lose points for not showing your bigrams and your friends' responses.
   - You also have a new part (c): Based on your data, describe how this modeling is similar to or different from n-gram language modeling.

3. Do question #2a in chapter 2 of the textbook (p. 65), regarding your own SOUNDEX algorithm.

4. Pretend we have a bigram array, as in the given table, where the first letter of the bigram is given in the vertical letters, and the second letter is given across the top.

		second		
		g	h	i
first	g	?	?	?
	h	?	?	?
	i	?	?	?

   (a) Make this into a positional bigram array, namely one which captures the position "end of word". Provide a word which justifies each 1 you put in the chart.

   (b) Each of the 1s you put in the chart may not be equal, in that they may not be equally likely. Discuss your intuitions as to how the chart would change were you to put in frequencies or probabilities. (Feel free to provide such a hypothetical chart.)

5. A user types in *bae* when they meant to type *able*. Draw the directed graph and describe how minimum edit distance is calculated. (Adapted from question #3 in chapter 2 (p. 66).)

6. Consider the misspelling *loe*, and assume our edit distance calculations have insertions, deletions, substitutions, **and** transpositions. Describe how probabilities are used to rank *ole*, *aloe*, *floe*, *lore*, *lode*, and any other words you think of. Do you have an intuition as to which should be highest, and why?

   (Be sure to discuss both kinds of probabilities involved in the noisy channel model. And feel free to suggest other relevant probabilities, if you think there are ones to consider.)

7. **Bonus:** We don't have an "honors" version of this course, but for those who want more of a challenge, give a line-by-line explanation of Peter Norvig's 21-line spelling correction system: `http://norvig.com/spell-correct.html`

   Then, choose (and possibly) tweak the implementation in the programming language of your choice and test its accuracy. Try different texts to train the spelling corrector, too: can you, e.g., develop a spelling corrector which works really well in a given domain?