CS224N/Ling 237 Homework #5

Due: Wed, May 5, 2004, 5PM

This assignment covers topics related to POS tagging, and HMMs starting with a couple of written questions, then moving to doing some practical language model construction (using the icopost package). FOR THIS HOMEWORK, ALL QUESTIONS MUST BE DONE INDIVIDUALLY.

Written questions

1. **POS HMMs (4 pts).**

   Suppose I am doing HMM POS tagging for the following string over the tag set {Det, N, V, P, Inf}, where 'Det' = determiner, 'N' = noun, 'V' = verb, 'P' = preposition, and 'Inf' is the 'to' that comes before English infinitive verbs:

   a ticket to ride

   And suppose I have the following state transition chart (where the from states are the rows, and to states are the columns):

	Det	N	V	P	Inf
Det	0	0.95	0.01	0.04	0
N	0.05	0.25	0.3	0.3	0.1
V	0.35	0.2	0.05	0.25	0.15
P	0.5	0.35	0.05	0.1	0
Inf	0	0	1	0	0

   Finally, I have the following (partially specified) emission probabilities (using a model where the observed symbol emitted at state t_i is dependent only on state X_i, and not on any preceding states):

Det	$\rightarrow$	a	0.4
N	$\rightarrow$	ticket	0.01
	$\rightarrow$	ride	0.008
V	$\rightarrow$	ticket	0.04
	$\rightarrow$	ride	0.08
P	$\rightarrow$	to	0.1
Inf	$\rightarrow$	to	1.0

Assume that the starting state is always Det. Furthermore, I have collected for each word in the above sentence the overall probabilities of occurrence as each POS, as follows:

	N	V		Inf	P		Det
ticket	0.8	0.2	to	0.5	0.5	a	1.0
ride	0.5	0.5					

Now suppose I want to compare the probabilities of the two following tag sequences:

(a) Det N Inf V
(b) Det N P N

Making use of the Markov assumption that the emission w_i is independent of state t_{i-1}, I calculate the probability of each sequence as follows:

$$\begin{aligned} P(a) &= P(t_1 = Det|w_1 = a) \times P(t_2 = N|t_1 = Det) \times P(t_2 = N|w_2 = ticket) \times P(t_3 = Inf|t_2 = N) \\ &\quad \times P(t_3 = Inf|w_3 = to) \times P(t_4 = V|t_3 = Inf) \times P(t_4 = V|w_4 = ride) \\ &= 1.0 \times 0.95 \times 0.8 \times 0.1 \times 0.5 \times 1.0 \times 0.5 \\ &= 0.019 \end{aligned}$$

$$\begin{aligned} P(b) &= p(t_1 = Det|w_1 = a) \times P(t_2 = N|t_1 = Det) \times P(t_2 = N|w_2 = ticket) \times P(t_3 = P|t_2 = N) \\ &\quad \times P(t_3 = P|w_3 = to) \times P(t_4 = N|t_3 = P) \times P(t_4 = N|w_4 = ride) \\ &= 1.0 \times 0.95 \times 0.8 \times 0.3 \times 0.5 \times 0.35 \times 0.5 \\ &= 0.01995 \end{aligned}$$

and conclude that (b) is the more likely tag sequence.

What is wrong with my calculation? Explain, show the proper calculation of probabilities for these two tag sequences, and discuss what the qualitatively important differences are (in terms of these particular tag probabilities).

2. **More Linguistic Issues (3 pts).**

   Another "word sense disambiguation" problem is sentence boundary detection. For a lot of text processing purposes, such as parsing, and semantic interpretation, we would like to divide texts into sentences. However, periods in English, and sometimes also question marks and exclamations, are ambiguous as to whether they represent the end of a sentence or not. For instance, in the text:

   > I went to talk with Prof. Manning who likes to to talk about linguistics, Java, Sydney, etc., but not about regrade requests. He said that I should look at item 4.3 again. And then we talked about the U.S.

   Some of the periods mark the ends of sentences and some don't. What features might one use to detect which punctuation marks are sentence boundaries (3 features is reasonable). Suggest a (reasonable) way in which they could be combined into a sentence boundary detection module. (Lecture on the week of April 26, 2004 had a discussion of features.)

3. **HMMs (10 pts).**

Consider the HMM with two states, X and Y, and an output alphabet $\{x, y\}$. The start-state probabilities, state transition probabilities and symbol emission probabilities are all unknown. This HMM is observed to give the output 'x' at $t = 1$, and 'y' at $t = 2$.

(a) We want to try to learn the probability parameters for this HMM from the output sequence observed. To do this, we can use the EM algorithm for parameter estimation. The particular case of the EM algorithm for HMMs is also known as the Forward-Backward algorithm. To use EM, we must start with some initial "guess" values of the parameters, and then run successive iterations of EM with the output sequence observed, to refine our estimates of the parameters. We stop iterating when the parameter values converge.

Use EM to estimate the parameters for this HMM, for each of the following three initial sets of starting parameters (where Π is the start state probabilities, A is the state transition probabilities, and B is the symbol emission probabilities). A good way to do this and see/learn what is going on is to make an Excel spreadsheet that does the calculations for one round of EM. If you then (by hand) copy across the new parameter values, the spreadsheet will then do the next round of EM.

Run EM for 2 iterations, or until the parameters converge, whichever comes first:

(i)

Π		
	X	1
	Y	0

A	X	Y
X	0	1
Y	1	0

B	x	y
X	1	0
Y	0	1

(ii)

Π		
	X	0.5
	Y	0.5

A	X	Y
X	0.5	0.5
Y	0.5	0.5

B	x	y
X	0.5	0.5
Y	0.5	0.5

(iii)

Π		
X	0.5	
Y	0.5	

A	X	Y
X	0.5	0.5
Y	0.5	0.5

B	x	y
X	0.9	0.1
Y	0.2	0.8

(b) What can you observe from these results about the effect of different initialization choices on the outcome of the values of the parameters in EM learning? Discuss with reference to the differences in the three results obtained above.

(c) Now assume that the data sequence is actually being generated by an HMM whose parameters are exactly the ones used in the initialization condition in (3a.i) above. However, when running EM for part (3a.iii), you would notice that not all of the parameters moved closer to the true values of the generating HMM. Explain why this is the case, and what would need to change so that the parameters all improve with successive iterations of EM on this HMM.

(d) Specify all other parameter settings that would assign the same likelihood to the observation sequence as the HMM whose parameters are exactly the ones used in the initialization condition in (3a.i) above.

(e) Does the class of HMM models we are considering define a probability distribution over all strings $s \in (x|y)^*$? (**Hint** Consider the probability mass assigned to strings of different lengths). If not, how can we extend the models to do so? In what applications might it be important that our model define a distribution over observation sequences of all lengths?

(f) Parameter tying is useful for reducing the number of parameters of a model and can be very helpful for improving performance. Consider tying the emission probabilities for states X and Y in the HMM model considered here (tying the parameters means adding in a constraint that they be equal, i.e., $P(x|X) = P(x|Y)$ and $P(y|X) = p(y|Y)$). Write out the expression for the likelihood of the sequence xy in terms of the parameters. Specify the parameters of a maximum likelihood HMM model (there may not be a unique one).

4. *Practical.* **POS tagging (8pts).**

We have installed in

/afs/ir/class/cs224n/src/icopost-0.9.1

a part of speech tagger by Ingo Schröder – well, there's actually a family of 3 part of speech taggers: an HMM tagger, a maximum entropy (loglinear model) tagger, and a transformation-based learning tagger, but we'll just use the trigram tagger. (Note: Icopost is now officially renamed Acapost and located at SourceForge. But we're using the Icopost version on the afs system.) There's some documentation for it in the html subdirectory. The current source and documentation (if you're curious) is at:

http://acopost.sourceforge.net/

We put a large amount of training text (extension .dat – this corresponds to what the documentation calls a "cooked" file), an English ngram and lexicon file in the data directory for use with the tagger. The data here comes from the Penn Treebank, and is tagged with the Penn Treebank tagset (see chapter 4 of M&S). You can use it to tag myfile.txt in your home directory with a command like the following if you are sitting in the data directory (see the documentation for details):

```
../bin/t3 -r 3 -v 3 english-wsj-train-0-18.ngram english-wsj-train-0-18.lex $HOME/myfile.txt
> $HOME/myfile.tagged
```

We also installed in

/afs/ir/class/cs224n/src/brill-tagger

Eric Brill's well-known transformation-based learning tagger (see M&S ch. 10 for discussion). If you go to the Bin_and_Data directory of it, you should be able to run it with the command:

```
./tagger LEXICON /afs/ir/class/cs224n/src/icopost-0.9.1/data/test-article.txt BIGRAMS
LEXICALRULEFILE CONTEXTUALRULEFILE > ~/article-tagged.txt
```

(where that should all be one long line). If you're in another directory, you need to give paths to the data files.

With these long command lines, you might want to define some aliases or little scripts to run the taggers!

Run both these taggers on the two test data files:

/afs/ir/class/cs224n/src/icopost-0.9.1/data/test-article.txt
/afs/ir/class/cs224n/src/icopost-0.9.1/data/test-novel.txt

Compare their output (you should be able to usefully use commands tr, diff, and wc).

(a) Where do they disagree? Give your judgment of which one is right in those cases, with some justification. (It isn't always easy to decide which is right: as well as the material in Chapter 3 of the textbook, you might look at:

/afs/ir/data/linguistic-data/Treebank/3/docs/tagguid1.ps

which particularly gives instructions for tagging with the Penn Treebank tag set.

(b) Are there any places that you notice where both are wrong?

(c) For at least two errors, give an explanation of why the tagger probably made a mistake (e.g., "this probably happened because *can* is most commonly used as a modal verb, and in this situation there isn't strong contextual evidence such as a preceding article to show that it is being used as a noun"). Are there any inexplicable errors?