CS224N/Ling 237 Homework #2

Due: May 6, 2002

This assignment is mainly written questions, with one involving POS tagging tools. The total is again 30pts.

1. (3pts) Smoothing probability models

   (a) In the absolute discounting model of smoothing, all non-zero MLE frequencies are discounted by a constant amount δ:

   Absolute discounting: If $C(w_n|w_1 \cdots w_{n-1}) = r$,

   $$P_{\text{abs}}(w_n|w_1 \cdots w_{n-1}) = \begin{cases} (r-\delta)/N & \text{if } r > 0 \\ \frac{(V-N_0)\delta}{N_0 N} & \text{otherwise} \end{cases}$$

   (Recall that V is the size of the vocabulary, N is the total number of times $w_1 \cdots w_{n-1}$ has been seen, and N_0 is the number of word types that were unseen after this context.) Show that this yields a probability model. In particular show that $\sum_{x \in V} P_{\text{abs}}(x) = 1$. What conditions are needed on δ?

   (b) Good-Turing estimation. Let the training text consist of N m-grams and let N_r be the number of m-grams that occurred exactly r times. Then,

   $$N = \sum_r r\, N_r$$

   The Good-Turing estimate P_{gt} for the probability of an m-gram w_r that occurred r times, $r > 0$, in the sample is

   $$P_{\text{gt}}(X = w_r) = \frac{r^*}{N} \qquad \text{where } r^* = (r+1)\frac{N_{r+1}}{N_r}$$

   We discussed in class how one needs to do some smoothing of this estimate to allow for places where N_r is 0, but let's just ignore that for this question.

   Assume now that there are N_0 m-grams that occurred 0 times in the training text. Calculate the total Good-Turing probability mass for all m-grams that did occur in the training text, and hence calculate the Good-Turing probability estimate for each m-gram that did not occur in the training text (that is, $P_{\text{gt}}(X = w_0)$), where w_0 is an m-gram that was absent from the training corpus). (We gave the answer in class, but want to see a derivation of it here.)

2. (4pts)

   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
ticket	0.8	0.2
ride	0.5	0.5

	Inf	P
to	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:

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

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

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).

3. (5pts) More linguistic issues

   (a) Briefly discuss the validity of the "one sense per discourse" constraint for different types of ambiguity (types of usages, homonyms, etc.). Discuss cases where the constraint is expected to do well and cases where it is expected to do poorly.

   (b) 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. Suggest a (reasonable) way in which they could be combined into a sentence boundary detection module.

   (c) The n-gram models we've shown make the $(n-1)$-order Markov assumption, i.e. that the distribution of words depends only on the previous $n-1$ words.

      i. Discuss (briefly) several ways in which this assumption is false for natural language. What properties of language will it not capture?

      ii. Despite the above, n-gram models are remarkably good at predicting the next word. Discuss why this might be. What information *is* in the previous word(s) that makes these models perform so surprisingly well?

4. (2 pts) Given the definition of cross-entropy in terms of the definitions of entropy and KL-divergence, show that the righthand side equality holds:

$$H(X, \mathrm{q}) \stackrel{\mathrm{def}}{=} H(X) + D(\mathrm{p} \parallel \mathrm{q}) = -\sum_{x} \mathrm{p}(x) \log \mathrm{q}(x)$$

5. (5 pts) Simplified Tongan

   (a) Simplified Tongan has 8 letters: *p, t, k, s, h, a, i, u*. Assuming that the language is just a random sequence of these letters, and that the letters have a uniform probability, then what is the per-letter entropy of Simplified Tongan (in bits)?

   (b) Closer statistical analysis suggests that the frequencies of the letters in Simplified Tongan are more correctly given as shown:

p	t	k	s	h	a	i	u
1/8	1/8	1/16	1/8	1/16	1/4	1/8	1/8

   Still assuming that the language is just a random sequence of letters, but with this overall distribution, what is the per-letter entropy?

(c) Even closer observation suggests that the language actually has syllable structure – in fact it is a strict sequence of CV syllables. The frequency of different syllables is shown in the table below:

	a	i	u	
p	1/8	1/8	0	1/4
t	1/8	1/16	1/16	1/4
k	1/16	0	1/16	1/8
s	1/8	1/16	1/16	1/4
h	1/16	0	1/16	1/8
	1/2	1/4	1/4	

Assuming that the language consists of a random sequence of independently generated syllables with the probabilities shown, what is the per-syllable entropy rate? Given that each syllable is two letters, what is the per-letter entropy rate? (That is, it is half the per-syllable entropy.)

(d) The above table can be interpreted as showing the joint distribution of two variables:

C: the syllable begins with the consonant c
V: the syllable ends with the vowel v

with marginal distributions as shown. What is the mutual information $I(C; V)$?

(e) An intrepid fieldworker brought back a real sample of Simplified Tongan:

tapituku titakapa tasasihu hapapisu

Using this 'text', calculate an estimate for the per-letter cross entropy between Simplified Tongan and each of the three models mentioned above. Does the cross entropy measure confirm the intuition that the syllable-based model is the best model of the language (of the ones we have considered)?

6. (5 points) 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 (2a.i) above. However, when running EM for part (2a.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.

7. (6pts) *Practical*. POS tagging. 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. There's some documentation for it in the html subdirectory or at:

http://nats-www.informatik.uni-hamburg.de/~ingo/icopost/

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. 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 from last time like tr, diff, and wc).

- 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.
- Are there any places that you notice where both are wrong?
- 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?
- What tagging accuracy do you get for the two taggers on these (small!) samples?