Solutions to CS224N/Ling 237 Homework #2

Wednesday, 25 April 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.)

Solutions:

(a) **Proof:** We know that the total number of instances is N.
If we were not doing any smoothing, then the MLE for items seen r times would be $\frac{r}{N}$ and the sum of all of these MLEs would be 1.
We are instead using $\frac{r-\delta}{N}$, so we are subtracting $\frac{\delta}{N}$ from each seen item.

There are $V - N_0$ items that have been seen. Therefore, we are subtracting a total of $(V - N_0) \times \frac{\delta}{N}$ from the probability mass of the seen words.

The probability of any unseen word is $\frac{(V-N_0)\delta}{N_0 N}$. Since there are N_0 unseen words, the total probability mass for the unseen words is $\frac{(V-N_0)\delta}{N}$. Since this is the probability mass that was subtracted from the seen words, the probability mass for all words, seen and unseen, is 1.

δ must be greater than or equal to 0, since we are subtracting $\frac{\delta}{N}$ from each seen item, and we do not want to subtract negative probability mass, as that would cause the probability mass for the seen items to be greater than 1. Further more, δ must be less than 1, otherwise the probability of a word seen once would be non-positive.

(b) **Proof:** Now, for the previously seen n-grams ($f(x_1 \cdots x_n) > 0$):

$$\sum_{x_1 \cdots x_n} g(x_1 \cdots x_n) = \sum_{x_1 \cdots x_n} \frac{(f(x_1 \cdots x_n) + 1) N_{f(x_1 \cdots x_n)+1}}{N_{f(x_1 \cdots x_n)}}$$

Partition these $x_1 \cdots x_n$ into groups by frequency, $1 \le r \le N$. Then, since there are N_r n-grams in the group of frequency r,

$$\begin{aligned}
\sum_{x_1 \cdots x_n} g(x_1 \cdots x_n) &= \sum_{r=1}^{N} \frac{(r+1)N_{r+1}}{N_r} N_r \\
&= \sum_{r=1}^{N} (r+1) N_{r+1} \\
&= \sum_{r=2}^{N} r N_r \\
&= N - N_1
\end{aligned}$$

since $N = \sum_{r=1}^{\infty} N_r$. Therefore the probability mass given to previously seen n-grams is $(N - N_1)/N$. So, the probability mass given to previously unseen n-grams is N_1/N, and so the probability estimate for each is $\frac{N_1}{N_0 N}$.
On this problem, copying an equation from the book and/or not showing all the steps would cost some credit.

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).

Solution: For an HMM, the probability of a given state sequence and observation sequence is decomposable into the probability of the state sequence times the probability of the observation sequence, given the state sequence. So instead of including $P(t_i|w_i)$, I should be calculating $P(w_i|t_i)$. (Also, I had left out $P(w_1 = a|t_1 = Det)$, but this mistake is more forgivable since for what we're interested in, the comparison of probabilities for these sequences, it's a wash.) My calculations should be:

$$
\begin{aligned}
P(a) &= P(w_1 = a|t_1 = Det) \times P(t_2 = N|t_1 = Det) \times P(w_2 = ticket|t_2 = N) \\
&\quad \times P(t_3 = Inf|t_2 = N) \times P(w_3 = to|t_3 = Inf) \\
&\quad \times P(t_4 = V|t_3 = Inf) \times P(w_4 = ride|t_4 = V) \\
&= 0.4 \times 0.95 \times 0.01 \times 0.1 \times 1.0 \times 1.0 \times 0.08 \\
&= 0.0000304
\end{aligned}
$$

$$
\begin{aligned}
P(b) &= P(w_1 = a|t_1 = Det) \times P(t_2 = N|t_1 = Det) \times P(w_2 = ticket|t_2 = N) \\
&\quad \times P(t_3 = P|t_2 = N) \times P(w_3 = to|t_3 = P) \\
&\quad \times P(t_4 = N|t_3 = P) \times P(w_4 = ride|t_4 = N) \\
&= 0.4 \times 0.95 \times 0.01 \times 0.3 \times 0.1 \times 0.35 \times 0.008 \\
&= 0.0000003192
\end{aligned}
$$

When the correct probabilities are put in, (a) comes out clearly as the more probable state sequence. This result is basically due to the fact that for the probability matrices we have here, a greater proportion of V's will have "ride" as their emission than will N's, even though "ride" appears equally often as V and N (i.e. there were more N's in the training corpus than V's); and "to" is the only possible emission for Inf but one of many P's, even though Inf appears less often than P.

Some people incorrectly left in the $P(t_i|w|i)$ tags in anyway; counterintuitively, these conditional probabilities don't belong at all. Also, note that the probabilities given here in the solution are of the form $P(\text{Tags}|\text{Words})$. If you calcuated $P(\text{Tags,Words})$ instead, this was okay.

See Manning & Schütze, p. 379, for more discussion of the mistake to be corrected in this problem.

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?

Solutions:

(a) In general, the answer that we are looking for here is some notion that the one-sense-per-discourse constraint does well for homonyms, which are semantically unrelated, but cannot be expected to work well in the case of polysemy. If a word's senses are related, then they are likely to appear in the same document, so the one-sense-per-discourse constraint will fail

(b) Features that may help us to determine sentence boundaries include whether the previous word is a known abbreviation or a number, and whether the following word begins with a capital letter. We can divide common abbreviations into abbreviations that tend to precede a capitalized word (such as Mr. or Capt.) and abbreviations that do not tend to precede a capitalized word (such as etc.). For more information, see Manning & Schütze, Section 4.2.4.

(c) A common approach to this problem was to examine n-grams in terms of their performance in capturing dependencies on *topic*, *syntax*, and *collocations.* This was fine.

   i. The most obvious property of language that the linear Markov assumption fails to capture is hierarchical structure, e.g. phrasal syntax. Another more subtle issue with ngrams is that, because they consider words to be the only linguistic object, we lose all notions of word classes, parts of speech, features on words, morphology, and so on. Note that a few people said that n-grams fail to capture words' dependencies on the *following* words. This isn't generally true, because probabilities of the form $P(w_{i+1}|w_i)$ implicitly contain the information contained in $P(w_i|w_{i+1})$– that is, a word serves to condition probabilities of subsequent, which captures its dependency on those words.

   ii. Although we're explicitly modeling very little linguistic structure, there are quite a few important effects that are implicitly, if not compactly, encoded in word adjacency statistics. Because most selection/dependencies between words *are* local, and because much of phrase structure *is* local, we will learn a lot. For instance, although we will fail to learn classes like noun, adjective, etc., our model will learn an extensional version of facts such as "adjectives and nouns follow determiners", "nouns can follow adjectives", etc. Commonly, things like subject verb dependencies will be captured perfectly well too (both syntactic agreement, and semantic selection) – only not in cases when there are intervening adverbial phrases, etc. Moreover, we capture a lot of information about word association, collocations, and idioms that are not captured by the usual sort of linguistic phrase structure grammar model. Even for things like definiteness effects (choice of article *a/the/*$\emptyset$), although we appear to have no knowledge of the discourse, we can often do quite well in predicting this based on what the noun is.

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:

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

Solutions:

We want to show that

$$H(X) + D(\mathrm{p} \parallel \mathrm{q}) = -\sum_x \mathrm{p}(x) \log \mathrm{q}(x)$$

We know that

$$H(X) = -\sum_x \mathrm{p}(x) \log \mathrm{p}(x)$$

and

$$D(\mathrm{p} \parallel \mathrm{q}) = \sum_x \mathrm{p}(x) \log \frac{\mathrm{p}(x)}{\mathrm{q}(x)}$$

Therefore,

$$\begin{aligned} H(X) + D(\mathrm{p} \parallel \mathrm{q}) &= -\sum_x \mathrm{p}(x) \log \mathrm{p}(x) + \sum_x \mathrm{p}(x) \log \frac{\mathrm{p}(x)}{\mathrm{q}(x)} \\ &= -\sum_x \mathrm{p}(x)[\log \mathrm{p}(x) - \log \frac{\mathrm{p}(x)}{\mathrm{q}(x)}] \\ &= -\sum_x \mathrm{p}(x)[\log \mathrm{p}(x) - [\log \mathrm{p}(x) - \log \mathrm{q}(x)]] \\ &= -\sum_x \mathrm{p}(x) \log \mathrm{q}(x) \end{aligned}$$

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)?

Solutions:

(a) We will sum over all of the letters, l

$$
\begin{aligned}
H(L) &= -\sum_l [\mathrm{p}(l) \log \mathrm{p}(l)] \\
&= -\sum_l [\frac{1}{8} \log \frac{1}{8}] \\
&= -\log \frac{1}{8} \\
&= 3
\end{aligned}
$$

Therefore, the per-letter entropy of Simplified Tongan is 3 bits.

(b)

$$
H(L) = -\sum_l [\mathrm{p}(l) \log \mathrm{p}(l)]
$$

$$\begin{aligned}
&= -[5 \times \frac{1}{8} \log \frac{1}{8} + 2 \times \frac{1}{16} \log \frac{1}{16} + \frac{1}{4} \log \frac{1}{4}] \\
&= \frac{15}{8} + \frac{1}{2} + \frac{1}{2} \\
&= \frac{23}{28} \\
&= 2.875
\end{aligned}$$

(c)

$$\begin{aligned}
H(S) &= -\sum_s [\mathrm{p}(s) \log \mathrm{p}(s)] \\
&= -4 \times \frac{1}{8} \log 18 + 8 \times \frac{1}{16} \log \frac{1}{16} \\
&= \frac{12}{8} + \frac{32}{16} \\
&= 3.5
\end{aligned}$$

The per-syllable entropy is 3.5 bits, so the per-letter entropy is 1.75 bits.

(d)

$$I(C;V) = H(C) + H(V) - H(C,V)$$

$$\begin{aligned}
H(C) &= -\sum_c [\mathrm{p}(c) \log \mathrm{p}(c)] \\
&= -2 \times \frac{1}{8} \log \frac{1}{8} + 3 \times \frac{1}{4} \log \frac{1}{4} \\
&= \frac{6}{8} + \frac{6}{4} \\
&= 2.25
\end{aligned}$$

$$\begin{aligned}
H(V) &= -\sum_v [\mathrm{p}(v) \log \mathrm{p}(v)] \\
&= -2 \times \frac{1}{4} \log \frac{1}{4} + \frac{1}{2} \log \frac{1}{2} \\
&= \frac{4}{4} + \frac{1}{2} \\
&= 1.5
\end{aligned}$$

H(C, V) is the per-syllable entropy, which was calculated in the previous part. Therefore,

$$\begin{aligned}
I(C;V) &= 2.25 + 1.5 - 3.5 \\
&= 0.25
\end{aligned}$$

The mutual inforamation is 0.25 bits.

(e) Real Tongan has this distribution (t) for letter frequencies (based on the sample):

p	t	k	s	h	a	i	u
1/8	5/32	1/16	3/32	1/16	1/4	1/8	1/8

and this distribution (u) for syllable frequencies:

	a	i	u	
p	1/8	1/8	0	1/4
t	3/16	1/16	1/16	5/16
k	1/16	0	1/16	1/8
s	1/16	1/16	1/16	3/16
h	1/16	0	1/16	1/8
	1/2	1/4	1/4	

Model from part (a) : (Note that t(l) is the actual distribution, and p(l) is the model)

$$\begin{aligned} H(L,t) &= -\sum_l [t(l) \log \mathrm{p}(l)] \\ &= -\sum_l [t(l) \log \frac{1}{8}] \\ &= 3\sum_l t(l) \\ &= 3 \end{aligned}$$

Model from part (b) :

$$\begin{aligned} H(L,t) &= -\sum_l [t(l) \log \mathrm{p}(l)] \\ &= -[\frac{1}{8}\log\frac{1}{8} + \frac{5}{32}\log\frac{1}{8} + \frac{1}{16}\log\frac{1}{16} + \frac{3}{32}\log\frac{1}{8} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{4}\log\frac{1}{4} + \frac{1}{8}\log\frac{1}{8} + \frac{1}{8}\log\frac{1}{8}] \\ &= \frac{3}{8} + \frac{15}{32} + \frac{4}{16} + \frac{9}{32} + \frac{4}{16} + \frac{2}{4} + \frac{3}{8} + \frac{3}{8}] \\ &= \frac{23}{8} \\ &= 2.875 \end{aligned}$$

Model from part (c) :

$$\begin{aligned} H(S,t) &= -\sum_s [u(s) \log \mathrm{p}(s)] \\ &= -[2 \times \frac{1}{8}\log\frac{1}{8} + 8 \times \frac{1}{16} + \log\frac{1}{16} + \frac{1}{16}\log\frac{1}{8} + \frac{3}{16}\log\frac{1}{8}] \\ &= \frac{6}{8} + \frac{32}{16} + \frac{3}{16} + \frac{9}{16} \\ &= 3.5 \end{aligned}$$

The per-syllable cross-entropy is 3.5 bits, so the per-letter cross-entropy is 1.75 bits. Yes, this confirms our intuition that the syllable-based model is the best model of the language, since it has the lowest per-letter cross entropy with the text.

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.

Solutions:

(a) (i) Iteration 1:

Π	
X	1
Y	0

A	X	Y
X	0	1
Y	1	0

B	x	y
X	1	0
Y	0	1

Intermediate computations:

α	t=1	t=2
X	1	0
Y	0	1

β	t=1	t=2
X	1	0
Y	0	1

p[1]	X	Y
X	0	1
Y	0	0

p[2]	X	Y
X	0	0
Y	1	0

γ	t=1	t=2
X	1	0
Y	0	1

Re-estimated parameters:

Π	X	1
	Y	0

A	X	Y
X	0	1
Y	1	0

B	x	y
X	1	0
Y	0	1

EM has converged after one iteration, as none of the parameters have changed.

(ii) Iteration 1:

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

Intermediate computations:

α	t=1	t=2
X	0.5	0.25
Y	0.5	0.25

β	t=1	t=2
X	0.25	0.5
Y	0.25	0.5

p[1]	X	Y
X	0.25	0.25
Y	0.25	0.25

p[2]	X	Y
X	0.25	0.25
Y	0.25	0.25

γ	t=1	t=2
X	0.5	0.5
Y	0.5	0.5

Re-estimated parameters:

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

EM has converged after one iteration, as none of the parameters have changed.

(iii) Iteration 1:

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

Intermediate computations:

α	t=1	t=2
X	0.500	0.275
Y	0.500	0.275

β	t=1	t=2
X	0.405	0.100
Y	0.090	0.800

p[1]	X	Y
X	0.091	0.727
Y	0.020	0.162

p[2]	X	Y
X	0.056	0.056
Y	0.444	0.444

γ	t=1	t=2
X	0.818	0.111
Y	0.182	0.889

Re-estimated parameters:

Π	X	0.818
	Y	0.182

A	X	Y
X	0.158	0.842
Y	0.434	0.566

B	x	y
X	0.880	0.120
Y	0.170	0.830

Iteration 2:

Intermediate computations:

α	t=1	t=2
X	0.818	0.127
Y	0.182	0.624

β	t=1	t=2
X	0.632	0.120
Y	0.089	0.830

p[1]	X	Y
X	0.025	0.944
Y	0.003	0.027

p[2]	X	Y
X	0.004	0.024
Y	0.422	0.550

γ	t=1	t=2
X	0.970	0.028
Y	0.030	0.972

Re-estimated parameters:

Π	X	0.970
	Y	0.030

A	X	Y
X	0.030	0.970
Y	0.424	0.576

B	x	y
X	0.972	0.028
Y	0.030	0.970

EM has not converged after two iterations, so we stop here.

(b) These results show that the choice of initial parmeters greatly affects the way EM learns the parameter values. In the first two cases above, EM converged after a single iteration. This is because EM is a hill-climbing algorithm, and in both those cases, the initial parameters were such that they perfectly explained the output sequence, giving local maxima for EM to converge at. However, in the third case, further improvement could be had by tuning the parameters, so EM managed to improve the model and made some changes.

(c) In particular, the state transition probabilities from state Y moved further away from their "true" values. This can be explained by looking at the output sequence observed: "xy". When we assume that X usually generates x, and Y usually generates y, we can see that it seems likely that the machine is in state Y at time t=2. However, we do not see any evidence to show that it would move back to state X after that point, since there is no further data. So, from this output sequence, there is no reason to believe that the transition Y $\rightarrow$ X has a high probability. On the other hand, we know that there is a slight probability that the initial state was Y, while it is still more likely that the state at t=2 is Y, not X. Because of these two facts, there is actually a higher probability for the transition Y → Y than for Y → X. The parameters learnt using EM illustrate exactly this point. For the parameters to converge towards their ”true” values, a longer output sequence would help, even if just ”xyx”, to add weight to the Y → X transition.

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?

Solutions:
Article:

sentence fragment	Brill	Icopost
Arab leaders	NNP	JJ
a *fund-raising* reception	NN	JJ
a farm bill now *stuck* in	VBD	VBN
a *House-Senate* conference committee	NNP	JJ
from *there* , he was headed	EX	RB
the *Thune* reception	NNP	JJ
greeting the *Saudi* without	NNP	JJ
earlier this *month* , British Prime Minister	NNP	NN
in the Middle East *as* well as the next	NNP	RB
in the *anti-terrorism* campaign	JJ	NN
the *Arab* world	NNP	JJ

- For the tagging of *Arab*, in both ”Arab leaders” and ”Arab world”, the Icopost tagger is correct. These cases are tricky since ”Arab” can be a noun or an adjective. If it is a noun, it should be tagged NNP (proper noun), but if it is an adjective, it should be tagged JJ – the Penn tagset does not distinguish ”proper adjective”. If you think of another country or case where the noun and adjective form are distinguish, you’d say ”Canadian leaders” or ”the Jewish world”, which indicates that these usages are JJ.
- For the tagging of *fund-raising*, the Icopost tagger is correct, as the Penn Treebank tagging guidelines say that hyphenated modifiers should always be tagged as JJ.
- For the tagging of *stuck*, the Icopost tagger is correct. We can see this if we replace the word ”stuck” with the word ”taken/took”, which have different forms for VBN and VBD.
- For the tagging of *House-Senate*, the Brill tagger is correct. The Penn treebank tagging guidelines state that all ”hyphenated compound nouns acting as modifiers” should be tagged as proper nouns.
- For the tagging of *there*, the Icopost tagger is correct. In this context, ”there” is being used as an adverb – it refers to a location (you can ask the question ”where?”)
- For the tagging of *Thune*, the Brill tagger is correct, as ”Thune” is a person’s last name, and should be considered a proper noun.

- For the tagging of *Saudi*, the Brill tagger is correct, as the word occurs in a position typical of nouns (following a determiner and preceding a verb).
- For the tagging of *month*, the Icopost tagger is correct – there is no reason to believe that ″month″ should be treated as a proper noun.
- For the tagging of *as*, the Icopost tagger is correct – there is no reason to believe that ″as″ should be treated as a proper noun.
- For the tagging of *anti-terrorism*, the Brill tagger is correct, as the Penn Treebank tagging guidelines say that hyphenated modifiers should always be tagged as JJ.

Novel:

sentence fragment	Brill	Icopost
like a good father *(* having met	(	VBZ
very unreasonable applications *)* , prided	SYM	NN
for his *dear* daughters ' sake	JJ	RB
have given *up* any thing	IN	RP

- For the symbol *(*, the Brill tagger is correct, as the tagset provides a tag particularly for this symbol.
- For the symbol *)*, both taggers are wrong, as the tagset provides a tag *)* particularly for this symbol. The errors on these parentheses may be because of ″data preparation error″: the Penn Treebank recodes parentheses as -LCB- and -RCB- so they don't conflict with parentheses used to show phrase structure, and the taggers are perhaps expecting input in this form.
- For the tagging of *dear*, the Brill tagger is correct, as modifiers of a noun should be tagged as adjectives rather than adverbs.
- For the tagging of *up*, the Icopost tagger is correct. We can use the following tests from the Penn Treebank tagging guidelines:
  It can either precede or follow the object (″have given any thing up″)
  If we replace ″any thing″ with a pronoun, it must precede the word in question (*″have given up it″)
  It bears stress in clause-final postion (″Why don't you give up?″)

Errors made by both taggers were more difficult to find, but we expected some evidence that you had looked at the tags and thought about them. An example of an error is:

- ″*that* low profile″ is tagged as IN by both taggers, but it should be tagged DT.

The accuracies for the taggers varied from person to person, but we expected reasonable figures (above 90 percent) and for figures to agree with the answers in the previous sections.