Practice Exam Problems: HMMs and EM

Natural Language Processing (JHU 601.465/665)

Prof. Jason Eisner

1. You are an American politician, currently staying in a Baltimore hotel whose name you can't even remember. You have 30 days left to campaign before the election. You will spend each day in one state, and fly between states at night in your campaign plane. (Obviously you will not have time to visit all 50 states; but you may visit a state more than once.)

   You would like to be in certain states on certain dates. For example, on day 5 you would like to be in Chesterton, Indiana (for a photo op at the Wizard of Oz Festival). On day 6 you would like to be either in Minocqua, Wisconsin (for Beef-A-Rama) or in New York City (for Wigstock). And so on. On the other hand, you're almost out of money, so you prefer short flights to keep fuel costs low.

   Your political consultant has made a 50×30 grid, showing for each (state, day) pair the dollar benefit to you of being in that state on that day. Meanwhile, your budget manager has made a 50×50 grid, showing for each (state, state) pair the dollar cost of flying from the first state to the second.

   You need to compromise between the political consultant (who wants you at all the festivals, the wackier the better) and the budget manager (who would rather you save money by just staying here in Maryland all month). So you must find the optimal schedule, telling you where to go on each day. You will measure the goodness of a schedule by its net dollar worth: total political benefit minus total travel cost.

   (a) [2 points] What NLP task does this scheduling problem resemble?

   (b) [3 points] What NLP algorithm should you use for scheduling?

   (c) [4 points] In your analogy from (a), what kind of natural language objects correspond to:

   - states? _____
   - days? _____
   - dollars? _____

(d) [4 points] How many addition operations does the algorithm require on this dataset? (Treat subtraction as addition of a negative number.) It is important to explain your answer a bit, since there are a few slightly different ways of organizing the computation.

2. It is possible to use part-of-speech tagging methods for purposes other than part of speech. For example, you could try to identify noun phrases by tagging each word in a text as either NP (meaning "part of an NP") or X (meaning "other").

Suppose your tagger for this purpose is a bigram Hidden Markov Model of the sort that you have seen in class and on homework. You have used training data to estimate the following tag-to-word probabilities, among others. (These probabilities are unrealistic, but they will make the computation less painful.)

$p(\text{the} \mid \text{NP}) = 0.3$	$p(\text{the} \mid \text{X}) = 0$
$p(\text{will} \mid \text{NP}) = 0.1$	$p(\text{will} \mid \text{X}) = 0.2$
$p(\text{to} \mid \text{NP}) = 0.1$	$p(\text{to} \mid \text{X}) = 0.2$
$p(\text{win} \mid \text{NP}) = 0.1$	$p(\text{win} \mid \text{X}) = 0.2$
$p(. \mid \text{NP}) = 0$	$p(. \mid \text{X}) = 0.0789$

You have also collected the following tag bigram counts from training data. You can derive probabilities from these; don't do any smoothing for this problem.

$c(\text{NP NP}) = 666$	$c(\text{X NP}) = 333$
$c(\text{NP X}) = 333$	$c(\text{X X}) = 666$

Now you see the test sentence

Popeye has the will to win .

The correct tagging is

Popeye/NP has/X the/NP will/NP to/NP win/NP ./X

since "the will to win" is a noun phrase.

Your model may get this wrong, because "will," "to," and "win" are usually not part of a noun phrase. On the other hand, your model may get it right, because it is certain that "the" must be tagged as NP, and it knows that NP words are usually followed by more NP words.

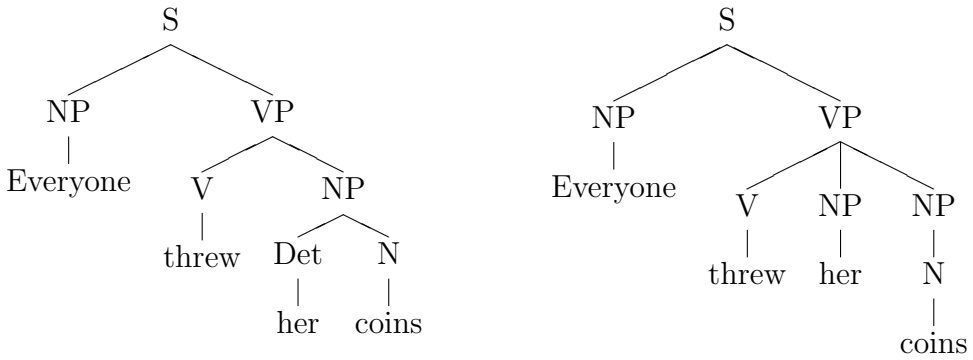
(a) [4 points] According to your model, which of the word tokens "will," "to," and "win" is most likely to have the tag NP *in this sentence*? You should be able to answer by thinking about the situation qualitatively, without doing any arithmetic.

Circle <u>one</u>: WILL TO WIN ALL EQUALLY LIKELY

(b) [11 points] Give the numerical probability, according to your model, that the token of "to" in this sentence has the tag NP. (Yes, it is possible to answer this even without knowing the probabilities for "Popeye" and "has.")

(*Hint:* Draw a trellis of possibilities for the last 5 tokens. You can be sure of the tags for "the" and "." Compute α and β values. You can save yourself some arithmetic if you remember that you only need to worry about *relative* path probabilities.)

3. Using a PCFG, you obtain two parses for this sentence:



The rules from your PCFG that were used above have the following probabilities:

_____	0.8	S → NP VP	
_____	0.5	NP → Det N	*left parse only*
_____	0.1	NP → N	*right parse only*
_____	0.5	VP → V NP	*left parse only*
_____	0.1	VP → V NP NP	*right parse only*
_____	0.001	NP → Everyone	
_____	0.001	V → threw	
_____	0.001	N → coins	
_____	0.03	Det → her	*left parse only*
_____	0.03	NP → her	*right parse only*

(a) [3 points] The left parse is more likely under this PCFG.
How many times more likely? _____

(b) [5 points] You would like to use the EM algorithm to reestimate some PCFG probabilities. The E step measures the expected count of each rule in this sentence. Write the expected counts (also known as "fractional counts") in the blank spaces to the left of each rule.

(c) [5 points] Owing to a catastrophic data loss, the above sentence is the only sentence in your corpus, so the M step has to reestimate the PCFG probabilities using only the expected counts above. What are the *new* probabilities of the four NP rules?

_____	NP → Det N	*left parse only*
_____	NP → N	*right parse only*
_____	NP → Everyone	
_____	NP → her	*right parse only*

(d) [6 points] This example only had 2 parses. In general, however, an n-word sentence might have exponentially many parses, so it is common to use the inside-outside algorithm to carry out the E step efficiently and find the expected counts.

Show that you know what the inside and outside probabilities are by computing them for a particular constituent, as follows:

inside probability: $\beta_{\text{NP}}(2,4) =$ _____

outside probability: $\alpha_{\text{NP}}(2,4) =$ _____

Use the original grammar probabilities shown at the start of the question, not any reestimated probabilities from your previous answers.

(e) [2 points] In class you learned the classic version of the inside-outside algorithm. Why can't you apply this version to the grammar above?

(f) [2 points] How long does the classic inside-outside algorithm take to compute the expected counts on an n-word sentence? (Assume a fixed grammar and answer in terms of n, using big-Oh notation.)

(g) [4 points] List the left corners of NP that can be determined from the rules given at the start of this problem. (Left corners were defined in HW3.)