Name: ____________________ Email: ____________________

Midterm Exam — Natural Language Processing — Prof. Eisner

Monday 28 March 2016, 3-4:30 pm
Total points: 80 Total minutes: 90

Note: **If you think a question is unclear or multiple answers are reasonable, please write a brief explanation of your answer, to be safe. Also show your work if you want wrong answers to have a chance at some credit: it lets us see how much you understood.**

1. You want to build a speech recognition system for *doctors* to dictate their notes, for example:[1]

```
On 4/4/14, the patient was taken to the operative theater.  An MRI
the following day showed gross total resection of the tumor.  Over
time her EVD was weaned and removed.  At the time of dismissal she
was at her neurological baseline.
```

You train a trigram language model with add-one smoothing, but it doesn't work very well because you only had a few thousand hospital records to train on.

(a) [2 points] You consider using backoff smoothing. This would tend to

[✓ INCREASE] DECREASE *(circle one)*

the bias of your estimates of the trigram probabilities, while

INCREASING [✓ DECREASING] *(circle one)*

the variance of those estimates.

(b) [2 points] A mysterious stranger on a street corner offers you millions of words of Internet chats where *patients* talk to one another about their health. You consider concatenating the chat room data to your hospital data. This would probably

[✓ INCREASE] DECREASE *(circle one)*

the bias of your estimates of the trigram probabilities, while

INCREASING [✓ DECREASING] *(circle one)*

the variance of those estimates.

[1] Example taken from bit.ly/1hIzUhu.

(c) [2 points] You try that idea, but it doesn't work very well for transcribing doctors' speech. Here's the problem. Let N_1 and N_2 be the total number of words in the hospital dataset and the chat dataset, respectively. N_1 is much smaller, so the most relevant data—the hospital records—are now only a tiny fraction of the concatenated dataset!

To fix this, you would like to "reweight" the data so that the hospital data have more influence in the combined model. Specifically, you want the hospital data to have weight α and the chat data to have weight $1 - \alpha$, where $0 \leq \alpha \leq 1$. For example, if $\alpha = 0.9$, then you will mostly trust the hospital data.

You will choose α by trying some different values of α and comparing the results on

TRAINING DATA [✓ DEVELOPMENT DATA] TEST DATA

(d) [2 points] Your first attempt to use α is an "interpolation" approach. You define

$$p(z \mid xy) = \alpha \cdot p_1(z \mid xy) \ + \ (1 - \alpha) \cdot p_2(z \mid xy)$$

where p_1 and p_2 are models trained separately on the two datasets.

You set $\alpha = 0.9$. However, this approach turns out to work badly. The trouble is that it *always* trusts p_1 more than p_2, but actually p_2 is more accurate in some contexts.

In particular, consider a context xy where $c_1(xy) = 0$ while $c_2(xy)$ is large. In this case, $p_1(z \mid xy)$ will be the ___Uniform___ distribution, which is not very accurate.

(e) [4 points] Concatenating the two datasets didn't have this problem because when estimating $p(z \mid xy)$ from the concatenated data, you were willing to consider any tokens of the bigram xy, from either corpus.

So, can you modify the concatenation approach so that when $\alpha > 0.5$, it *prefers* tokens that come from the hospital corpus, but will also consider tokens that come from the chat room corpus? You could call this approach "weighted concatenation."

Write a formula for an estimate of $p(z \mid xy)$, without backoff. When $\alpha > 0.5$, the estimate should be more influenced by the hospital corpus, even though that corpus is smaller.

$$p(z \mid xy) = \frac{\text{Alpha * c\_1(zxy) + (1 - Alpha) * (N1/N2) * c\_2(zxy) + 1}}{\text{Alpha * c\_1(xy) + (1 - Alpha) * (N1/N2) * c\_2(xy) + V}}$$

Hint: Your formula should still be related to add-one smoothing. It should mention α, N_1, N_2, and some n-gram counts $c_1(\ldots)$ and $c_2(\ldots)$.

(f) [3 points] Another approach would be backoff smoothing. You would like to estimate $p(z \mid x, y, \text{Speaker=doctor})$, in order to model what doctors say. But you are willing to back off to $p(z \mid x, y)$, which considers what *anyone* says.

The trouble with this idea is that $p(z \mid x, y, \text{Speaker=doctor})$ should also presumably back off to $p(z \mid y, \text{Speaker=doctor})$. It's not obvious how to make it back off to two different things at once.

With an ordinary trigram model, the backoff path was simple:

$$\begin{tikzcd} p(z \mid x, y) \arrow[r] & p(z \mid y) \arrow[r] & p(z) \end{tikzcd}$$

But here, we seem to want different kinds of backoff at once . . .

$$\begin{tikzcd} p(z \mid x, y, \text{doctor}) \arrow[rr] \arrow[dr] & & p(z \mid y, \text{doctor}) \arrow[rr] \arrow[dr] & & p(z \mid \text{doctor}) \arrow[dr] & \\ & p(z \mid x, y) \arrow[rr] & & p(z \mid y) \arrow[rr] & & p(z) \end{tikzcd}$$

In a phrase, what is a reasonable way to resolve this problem? (There is more than one reasonable answer.) ______________________________

Give a few details of your idea.

Using Chain Rule to expand p(z,x,y,doctor) then we can backoff to the estimations that make most sense and for terms that can backoff to multilple terms we can keep a weighted average of their native counts

Also Log Linear Models work

2. The following sentences use a "dummy subject" because the semantic representation doesn't call for a subject:

- It snowed.
- It rained all day.

Here are some other sentences that can be rephrased—their long subject moves after the verb, and a dummy subject is used instead. The rephrased version often seems easier to understand.

- Once upon a time, [three little pigs] were sitting at home.
  ⇒ Once upon a time, there were [three little pigs] sitting at home.
- [To chew these candies] is tough.
  ⇒ It is tough [to chew these candies].
- [To find oxygen on Mars] would be surprising.
  ⇒ It would be surprising [to find oxygen on Mars].
- [To attempt this question] takes guts.
  ⇒ It takes guts [to attempt this question].
- [To veto those bills] frustrated the president.
  ⇒ It frustrated the president [to veto those bills].

The act of rearranging a sentence's phrases is called a "syntactic transformation." Starting in 1957, Noam Chomsky proposed an "transformational grammar" approach to syntactic description, which combined a CFG with explicit syntactic transformations that post-processed the syntax tree. But in this class, we have described the same syntactic constructions without any transformations, just by using a fancier CFG with attributes, slashed categories, etc.

There is another way to transform some of these sentences:

- [These candies] are tough [to chew].
- [Oxygen] would be surprising [to find on Mars].
- [This question] takes guts [to attempt].
- [Those bills] frustrated the president [to veto].

This transformation was studied by Rosenbaum (1967), who named it "*tough*-movement" because examples like the first one involve the adjective *tough*.

(a) [3 points] The text above (within question 2) actually includes a sentence with a dummy subject. The sentence is part of the explanation, not an example in a bullet point. Underline that sentence, and circle the dummy subject!

(b) [3 points] Similarly, the text above (within question 2) includes a sentence that uses *tough*-movement. The sentence is part of the explanation, not an example in a bullet point. Underline that sentence, and circle the adjective that is involved.

(c) [8 points] Consider all of the sentences above. Think carefully about how to generate *tough*-movement sentences directly from a CFG for English (rather than by transforming some other form of the sentence). To illustrate your solution, draw your recommended parse for the following sentence. Choose your other nonterminals carefully as well, with appropriate VP attributes to handle the tenses of the verb phrases.

```
Oxygen   would   be   surprising   to   find   on   Mars
```

3. You find an injured English sentence! The 4th word's vowels have been cut out:

   `I played the bgl at camp .`

   You would like to heal the sentence, but was the 4th word originally `bagel`, `beagle`, `beguile`, or `bugle`? Let's suppose that all of these words have the same unigram probability. Note that `beguile` is a verb and the rest are nouns.

   (a) [1 point] `bugle` is the most probable choice in this context. A well-trained n-gram language model will tend to figure this out, but only if $n \geq$ 3.

   (b) [1 point] `beguile` is the least probable choice in this context. A well-trained n-gram language model will tend to figure this out, but only if $n \geq$ 2.

   (c) [3 points] You can also use a PCFG as a language model, because it assigns a probability to every string of words. However, not all English PCFGs would figure out that `bugle` is the most probable choice in this context. What *kind* of English PCFG would tend to figure that out?

   That Natively predicts word counts to give empirical probabilities.

   (d) [1 point] You can even use a CFG to solve this problem, to some extent. Which choice would an English CFG tend to reject as impossible in this context? beguile

4. Now you find a 12-word sentence in even worse condition. *All* of its vowels have been cut out. (It is still 12 words long, as none of the words consisted entirely of vowels.)

   `th ncst lngst amng s s frm the sm lngtd s blgrd`

   You have a PCFG for English, in Chomsky Normal Form. To understand this injured sentence, you would like to parse it. However, you are not sure what the words are! There are several possibilities at each position:

th	ncst	lngst	amng	s	s	frm	the	sm	lngtd	s	blgrd
the	incest	linguist	aiming	as	as	farm	the	same	elongated	as	belgrade
	nicest	longest	among	ease	ease	firm		seam	longitude	ease	beleaguered
				is	is	forum		seem		is	
				sea	sea	frame		some		sea	
				see	see	from		sum		see	
				so	so			sumo		so	
				us	us					us	

   So there are $1 \times 2 \times 2 \times 2 \times 7 \times 7 \times 5 \times 1 \times 6 \times 2 \times 7 = 164640$ possible sentences. You would rather not determine each one's probability by parsing it separately, which would be slow.

   Fortunately, a probabilistic CKY parser is designed to sort through an exponential number of possibilities and pick the highest-probability one, which is what you need!

Usually, the parser considers all trees whose leaf sequence *equals* a sequence of English words. This time, you want to consider all trees whose leaf sequence is *compatible with* the sequence of injured words.

For example, a parse of `the nicest longest among ease ease firm the seam longitude ease` would be a possible interpretation of the injured sentence—although such a tree would probably have very low probability and not win. (Look at the end of this question to see the winning tree.)

(a) [4 points] You are given a standard implementation of a probabilistic CKY parser. This parser expects as input a sequence of words, $w_1, w_2, \ldots w_n$. It handles the input as follows:

```
for $k := 1$ to $n$ :                                  ▷ initialize the diagonal of the chart
  for all rules $X \to w_k$ in the grammar :           ▷ by looking at the input words
    chart$[X, k-1, k] := p(X \to w_k \mid X)$
```

But for your situation, the input will be a sequence of *sets* of possible words, $W_1, W_2, \ldots W_n$. (In the example, $n = 12$, and $W_3 = \{$`linguist`, `longest`$\}$.)

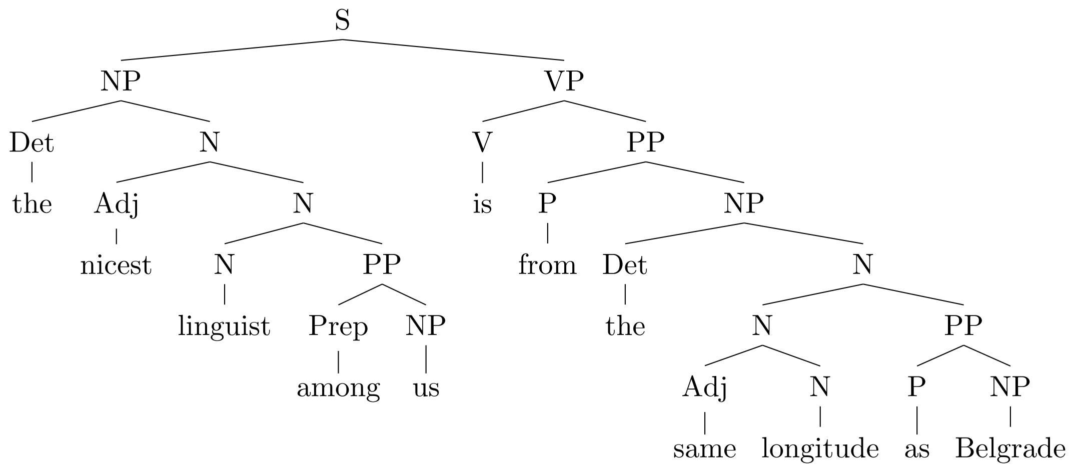
Modify the code so that it handles this case.

(b) [4 points] The probabilistic CKY parser first uses a bottom-up algorithm to find the probability of the most probable tree.

Then it has to actually print the most probable tree. In the standard implementation, printing the *leaves* is quite easy because they are just the input words. However, in your situation, the i^{th} leaf of the tree will be some word $w_i \in W_i$. You are very interested in printing these correctly, because the leaf sequence $w_1, w_2, \ldots, w_n$ will actually be a good guess about the original sequence of English words before the sentence was injured.

What change do you need to make to the standard implementation? Explain briefly but precisely.

(c) [5 points] When you run your modified parser, it finds that the most probable parse is is as follows:



Let **w** = `the nicest linguist among us is from the same longitude as Belgrade`. Can you conclude that **w** is the most probable sentence (according to the PCFG) that is consistent with the injured input $W_1, \ldots, W_{12}$?

YES [✓ NO] *(circle one)*

The PCFG probability of generating this tree is

$<$ [✓ $\leq$] $=$ $\geq$ $>$ *(circle one)*

the PCFG probability of generating the sentence **w**.

5. Consider again the injured sentence **x** from the previous problem, and its possible healed version **w**:

x = `th ncst lngst amng s s frm the sm lngtd s blgrd`
w = `the nicest linguist among us is from the same longitude as Belgrade`

We don't necessarily need a PCFG or even an n-gram model to choose **w**. Potentially, another way is to define a conditional log-linear model $p(\mathbf{w} \mid \mathbf{x})$, where **w** is an English sentence and **x** is a version with the vowels removed.

Let's define such a model. Recall that

$$p(\mathbf{w} \mid \mathbf{x}) = \frac{u(\mathbf{x}, \mathbf{w})}{Z(\mathbf{x})}$$

where $Z(\mathbf{x})$ is a summation.

(a) [2 points] For our particular example input **x** above, how many summands does the denominator $Z(\mathbf{x})$ have? ____________

(b) [2 points] What kind of data would you need to train the parameters of this conditional log-linear model?

(c) [2 points] Where could you obtain such data? (Assume that you have only seen a few injured sentences, even though a vowel-deleting computer virus is spreading. You want to train your model *now*, before the virus is widespread.)

(d) [2 points] What function would you maximize to train the parameters of the model? Is this function guaranteed to be concave?

(e) [4 points] Write a plausible expression for the numerator $u(\mathbf{x}, \mathbf{w})$. You will have to think of features. You can assume that both $\mathbf{x}$ and $\mathbf{w}$ have length n.

Since each w_i is a real English word, you can assume that it has an embedding as a d-dimensional vector; you can write that vector as $\vec{v}(w_i)$.

6. [3 points] If an event occurred r out of k times, Good-Turing smoothing estimates its probability as

(a) $\frac{r}{k} \cdot (1 - \sum_{s=1}^{\infty} \frac{N_s}{N_r})$ (b) $\frac{N_{r+1}(r+1)}{N_r r} \cdot \frac{N_0}{k}$ (c) $\frac{r+1}{k} \cdot \frac{N_{r+1}}{N_r}$ (d) none of these

7. (a) [2 points] You want to translate from French to English. Use Bayes' Theorem to re-express the following:

   $p(\textit{English} = e \mid \textit{French} = f) =$ ____

   (b) [3 points] Label the following three things in your answer above: (1) *prior probability*, (2) *posterior probability*, (3) *likelihood.*

   (c) [2 points] One way to get a model of translation is to build separate models for the parts of this expression. What is this approach called? ________________________

8. Here are the first few columns of an Earley's algorithm table. The grammar and input sentence are not shown, but you can get some information about them by looking at the table.

0	**1**	**2**	**3 . . .**
0 ROOT → . S	0 NP → the . Holy Grail	_ N → baby .	2 P → on .
0 S → . NP VP	0 DET → the .	_ NP → DET N .	⋮
0 NP → . DET N	0 NP → DET . N	_ S → NP . VP	
0 NP → . DET ADJ N	0 NP → DET . ADJ N	_ NP → NP . PP	
0 NP → . NP PP	1 N → . adult	_ VP → . V NP	
0 NP → . John	1 N → . teenager	_ VP → . VP PP	
0 NP → . Mary	1 N → . child	_ PP → . P NP	
0 NP → . the Holy Grail	1 N → . baby	_ V → . ate	
0 DET → . a	1 ADJ → . big	_ V → . slept	
0 DET → . the	1 ADJ → . little	_ V → . pooped	
0 DET → . every		_ P → . of	
		_ P → . on	
		_ P → . with	

   (a) [3 points] What are the first three words of the sentence?

   (b) [4 points] In column 2, fill in the blanks with numbers.

   (c) [3 points] Circle all of the entries that would be explicitly blocked by a "left corner" trick for 1-word lookahead. Then cross out those entries *and* all other entries that as a result would never get created.

Scratch Paper

Name: ______________________ Email: ______________________

Midterm Exam — Intro to NLP (JHU 600.465) — Prof. Eisner

Monday 20 October 2003, 2 pm

Total points: 90 (+ 10 extra credit)   Total minutes: 50

***Note:* If you think a question is unclear or multiple answers are reasonable, you can write a brief explanation of your answer, to be safe.**

1. [4 points] In the 2002 movie of the musical *Chicago*, the jail matron, "Mama" Morton, tells murderess Roxie Hart that her new lawyer is the best in town:

   MAMA: Billy Flynn's been to court 47 times and never lost a case.
   So now your odds are 47 to 1. (approximate quote—I didn't have time to rent the video!)

   Don't take Mama's word for it; work it out yourself. Under add-one smoothing, what is p(Roxie wins her case | Billy is her lawyer)?

2. (a) [4 points] If $p(x,y,z) > \frac{p(x,y)\cdot p(y,z)}{p(y)}$, then

   $$p(x \mid y,z) > p(\underline{\qquad\qquad} \mid \underline{\qquad\qquad}),$$

   which is POSSIBLE   IMPOSSIBLE (*circle one*).

   (b) [4 points] If the $>$ signs above are replaced by $=$ signs, then we say that ______ is conditionally independent of ______ given ______.

3. [5 points] Give an example of a garden-path sentence. Mark the point where a reader would pause in confusion during a self-paced reading task.

4. [5 points] Jack and Jill are walking up the hill when Jill says, "Look at that dog with no eyes!" Jack covers his eyes and asks, "Where?"

   To prevent Jack's misunderstanding of Jill's ambiguous sentence, what context-free rule in his grammar should be **lowered** in probability? In your answer, show the head features on the nonterminals.

5. Actually (true story), I was hiking downhill with my parents when we encountered a sweaty couple coming slowly up the hill with a panting dog. My mother made what sounded like one of the following remarks, but I wasn't sure which, since they were both grammatical and made sense:

   - `I think they're dog tired.` (sentence s_1)
   - `I think their dog tired.` (sentence s_2)

   This was a speech recognition problem for me.

   (a) [4 points] True or false: Using a bigram model with no smoothing (just naive, history-based, maximum-likelihood estimates), $p(s_1) > p(s_2)$ if and only if

   $$\frac{c(\texttt{think they're}) \cdot c(\texttt{they're dog})}{c(\texttt{they're})} > \frac{c(\texttt{think their}) \cdot c(\texttt{their dog})}{c(\texttt{their})}$$

   TRUE FALSE (*circle one*)

   (b) [6 points] Let s_3 be the sentence `Hello!` I was pretty sure my mother didn't say s_3 (it didn't *sound* like she did). Why would a good statistical speech recognizer rule out s_3? (*circle one*)

   (A) Because $p(s_3)$ is much smaller than $p(s_1)$ and $p(s_2)$.
   (B) Because $p(s_3)$ is much bigger than $p(s_1)$ and $p(s_2)$.
   (C) Because it would make its decision by comparing quantities other than $p(s_1)$, $p(s_2)$, and $p(s_3)$.
   (D) Because it would use well-smoothed estimates, not unsmoothed ones.
   (E) An speech recognizer with an n-gram language model could not rule out s_3. It would take a more sophisticated language model whose probabilities were conditioned on the social context (meeting on the hiking trail), not just on the past $n-1$ words.

6. (a) [5 points] Binarize the following grammar fragment:

    S $\rightarrow$ NP VP
    NP $\rightarrow$ Det N
    NP $\rightarrow$ Det Adj N
    VP $\rightarrow$ V NP
    VP $\rightarrow$ V NP NP

   (b) [5 points] Consider the original, unbinarized grammar fragment printed above. You want to extend it to handle phrases with NP gaps (e.g., `what did` `Marge feed Homer`, or `whom did` `Marge feed donuts`). Write all the new rules you need of the following forms:

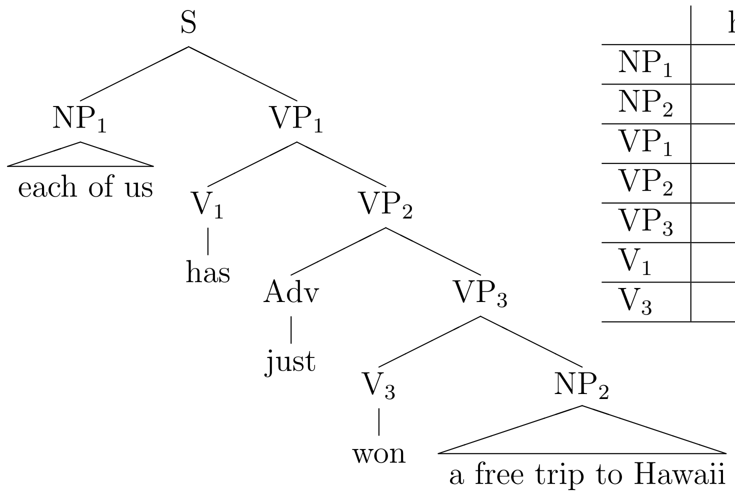
    S/NP $\rightarrow \ldots$
    NP/NP $\rightarrow \ldots$
    VP/NP $\rightarrow \ldots$

7. [9 points] You are building a trigram language model using **Witten-Bell backoff smoothing**. Your training data includes the trigrams `see the baby` and `see the kitty`, at least once each, but it does not include `see the corpse`.

   You observe that the smoothed estimate of $p($`baby` $\mid$ `see the`$)$ is exactly half of the unsmoothed estimate. Which of the following are necessarily true? *(circle all that apply)*

   (a) In training data, every instance of `see the` was followed by a different word.
   (b) The smoothed estimate of $p($`kitty` $\mid$ `see the`$)$ is also exactly half of the unsmoothed estimate.
   (c) The smoothed estimate of $p($`corpse` $\mid$ `see the`$)$ is exactly half of the smoothed backed-off estimate of $p($`corpse` $\mid$ `the`$)$.
   (d) none of the above

8. [11 points] Fill in the head, tense, and number features for the nonterminals in the tree below. Write your answers in the table provided. (N/A means that the constituent does not need to specify any value for the feature.)



	head	number	tense
NP$_1$			N/A
NP$_2$			N/A
VP$_1$			
VP$_2$		N/A	
VP$_3$		N/A	
V$_1$			
V$_3$		N/A	

9. In this question, you'll consider parsing under the following "permissive" probabilistic context-free grammar, which has just 2 rules, of equal probability:

   1/2 $A \to A\ A$
   1/2 $A \to a$

   The input is the 4-word sentence `a a a a`.

   (a) [5 points] Draw all parse trees for the input sentence. (If you like, you can leave out the letters A and a and just show the shape of each tree, since all the internal nodes are A and all the leaves are a.)

   (b) [8 points] Use the version of the CKY algorithm that finds the probability of the most probable parse. In the parse chart below, fill each cell with possible nonterminals and the maximum probability associated with each. (You don't have to show the backpointers that would help you recover the parse.)

	1	2	3	4
0				
1				
2				
3				

(c) [5 points] Suppose instead you used the version of CKY that builds up the *total* probability of all parses of the sentence. This is sometimes called the sentence's "inside probability." Is the inside probability always 1? Explain.

(d) [10 points] Running Earley's algorithm results in the finished parse chart below. Circle the **four** entries in the final column that were added to the chart more than once. (*Hint:* You could hand-simulate the creation of the final column, starting with its first entry. Or you could consider, for each element in the final column, all the ways that it could have been added; in this case it might help to think about all the ways that a CKY cell can be filled in.)

0	**1**	**2**	**3**	**4**
0 ROOT → . A	0 A → a .	1 A → a .	2 A → a .	3 A → a .
0 A → . A A	0 ROOT → A .	0 A → A A .	1 A → A A .	2 A → A A .
0 A → . a	0 A → A . A	1 A → A . A	0 A → A A .	1 A → A A .
	1 A → . A A	0 ROOT → A .	2 A → A . A	0 A → A A .
	1 A → . a	0 A → A . A	1 A → A . A	3 A → A . A
		2 A → . A A	0 ROOT → A .	2 A → A . A
		2 A → . a	0 A → A . A	1 A → A . A
			3 A → . A A	0 ROOT → A .
			3 A → . a	0 A → A . A
				4 A → . A A
				4 A → . a

10. [**extra credit**] Consider the following 3 corpora:

- A = the large Switchboard corpus of English telephone conversations
- B = `the aids issue is is a bit of a problem`
  (which is just the first sentence of the Switchboard corpus)
- C = `i saw snuffleupagus on tv`

Using `fileprob`, you train a trigram language model on either A or B, and test it on C. If you train on A, you get $\log_2 p(C) = -106$. If you train on B, you get $\log_2 p(C) = -25$.

(a) [3 **extra credit** points] What is the *perplexity per word* of the second model (the one trained on B), when tested on C? Ignore the special word EOC ("end of corpus").

(b) [7 **extra credit** points] The model trained on B appears to be better—it gave much higher probability to test data C.

But that doesn't make any sense. A is a much bigger and better training corpus. In fact, A included almost all the words in test data, whereas B included none of them. So how can B be less perplexed by the test data?

Explain the paradox. (*Hint:* It has to do with OOV words. Think about the size and meaning of $p(\text{OOV})$ under each model. Just think about add-1 smoothing to keep things simple; other smoothing methods would behave similarly.)