Name: _____ Email: _____

Natural Language Processing (600.465) — Final Exam

Johns Hopkins University (Prof. Jason Eisner)
Friday 6 May 2016, 2pm–5pm
Total points: 180 Total minutes: 180

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

1. [3 points] Draw a finite-state acceptor for the regular expression `[fee |fie |fo |fum ]+`, which accepts strings such as "`fee fie fo fum` " and "`fum fum fum` ". The arcs should be labeled with characters (not words). Use as few states as possible.

2. [10 points] This cartoon by Michael Maslin appeared in *The New Yorker*:

"So the 'ah' is always followed by a 'choo'?"

Suppose the patient said "ah" 3 times today, and always followed it by "choo." A bigram language model should predict the probability that the next "ah" will also be followed by "choo."

Assume that that the patient's total vocabulary size including OOV is 50 (must be a young patient!) and that the smoothed *unigram* probability $p(\text{choo})$ is $\frac{1}{1000}$.

What is the smoothed *bigram* probability $p(\text{choo} \mid \text{ah})$? Draw a line from each smoothing method to the answer it gives—make sure we can tell where your line goes. It is possible for different smoothing methods to connect to the same answer.

	1
	0
unsmoothed (maximum-likelihood estimate)	$3/4$
	$3/50$
add-one smoothing	$(3+1)/50$
	$(3+1)/(3+50)$
add-one smoothing with backoff	$3/(3+51)$
	$(3+1)/(3+51)$
Witten-Bell smoothing with backoff	$(3+\frac{1}{1000})/4$
	$(3+\frac{1}{1000})/(3+50)$
one-count smoothing with backoff	$(3+\frac{50}{1000})/(3+50)$
(you may have implemented this in HW6)	SOMETHING ELSE
	NOT ENOUGH INFORMATION GIVEN

no longer covered in this class

3. In homework 2, we met beavers who communicated with cries from the vocabulary $V = \{\texttt{bwa}, \texttt{bwee}, \texttt{kiki}\}$. Let's build a simple language model of beavers.

   First let's assume that each sentence consists of a single cry. Then the language model only has to specify a distribution over V. In a training corpus, you observe empirical probabilities

   $$\tilde{p}(\texttt{bwa}) = 0.3 \qquad \tilde{p}(\texttt{bwee}) = 0.2 \qquad \tilde{p}(\texttt{kiki}) = 0.5$$

   We'll use a log-linear model with three features, one per cry:

   $$p_\theta(\texttt{bwa}) = \tfrac{1}{Z}\exp\theta_{\texttt{bwa}} \qquad p_\theta(\texttt{bwee}) = \tfrac{1}{Z}\exp\theta_{\texttt{bwee}} \qquad p_\theta(\texttt{kiki}) = \tfrac{1}{Z}\exp\theta_{\texttt{kiki}}$$

   (a) [3 points] If you train the parameter vector θ to maximize likelihood, then $p_\theta(\texttt{bwee}) =$ _____

(b) [3 points] Give a parameter vector that would maximize likelihood. (Hint: Think about the gradient.)

$\theta_{\texttt{bwa}} =$ _____ $\theta_{\texttt{bwee}} =$ _____ $\theta_{\texttt{kiki}} =$ _____

(c) [2 points] The maximizer is not unique. Give a *different* parameter vector that would *also* maximize likelihood.

$\theta_{\texttt{bwa}} =$ _____ $\theta_{\texttt{bwee}} =$ _____ $\theta_{\texttt{kiki}} =$ _____

(d) [2 points] Which parameter vector would an L_1 regularizer prefer?
MY ANSWER FROM ii. MY ANSWER FROM iii. *(circle <u>one</u>)*

Suppose you simplify your model to the following:

$$p_\theta(\texttt{bwa}) = \tfrac{1}{Z} \exp \theta_{\texttt{bw}} \qquad p_\theta(\texttt{bwee}) = \tfrac{1}{Z} \exp \theta_{\texttt{bw}} \qquad p_\theta(\texttt{kiki}) = \tfrac{1}{Z} \exp \theta_{\texttt{kiki}}$$

This model has only has 2 parameters, since now `bwa` and `bwee` share a parameter $\theta_{\texttt{bw}}$.

(e) [3 points] Under the new model, the answer to question 3a is $p_\theta(\texttt{bwee}) =$ _____

Now we discover that beaver sentences can be of any length, so we need a distribution over sentences $x \in V^*$:

$$p(x) = \tfrac{1}{Z} \exp \left(\theta_{\texttt{bwa}}\, c_x(\texttt{bwa}) + \theta_{\texttt{bwee}}\, c_x(\texttt{bwee}) + \theta_{\texttt{kiki}}\, c_x(\texttt{kiki}) \right)$$

where $c_x(w)$ is the count of word w in sentence x. In our new training corpus of sentences, we again observe

$$\tilde{p}(\texttt{bwa}) = 0.3 \qquad \tilde{p}(\texttt{bwee}) = 0.2 \qquad \tilde{p}(\texttt{kiki}) = 0.5$$

Also, the average training sentence has length 4.0.

(f) [3 points] Note that our model doesn't have any features for sentence length or for EOS.

If you train the parameter vector θ to maximize likelihood, will the trained model correctly predict that sentences have length 4.0? Answer YES or NO, and explain your answer.

4. Let `Phone` be an FST that maps an uppercase letter to its corresponding phone digit:


```
(A|B|C):2 | (D|E|F):3 | (G|H|I):4 | (J|K|L):5 |
(M|N|O):6 | (P|R|S):7 | (T|U|V):8 | (W|X|Y):9
```

(a) [2 points] If `x` denotes a string of uppercase letters, write a regular expression that matches (only) the corresponding string of digits. For example, if `x` matches `NLP`, then your expression should match `657`.

(Your answer should contain `x` and `Phone`. Remember that `x` may have multiple characters but `Phone` only handles a *single* character.)

(b) [1 point] `Z` doesn't appear on the phone keyboard. So if $x =$ `ZEBRA`, what does your regular expression match?

(c) [4 points] You are choosing a phone number. You like hobbits, so you think `HOBBITS` would be a nice number. You also like jam and swimming, so you think `JAMSWIM` or `SWIMJAM` would be a nice number.

Actually, you like a lot of things. You define `Nice` to be a regular expression that matches all the words you like.

But you are afraid that your big sister might make fun of you if your phone number contains a substring like `8459`, because that could be taken to spell `UGLY`.

Actually, you realize a lot of words could be problems. You like snorkeling, but if you take the number `SNORKEL`, or `766 7535`, your sister might run after you on the street yelling "There goes `POOP 535`! Admit it, that's your phone number!"

So you define `Nasty` to be a regular expression that matches all the words you don't like.

You would like a **7-digit** phone number that **can** be spelled as a word sequence in `Nice*`, but where **none** of its substrings can be spelled as a word in `Nasty`.

Write a regexp that matches exactly the phone numbers that meet your requirements.

5. (a) [4 points] A palindrome is a string that is unchanged by reversal, such as `abekeba`. There is not an FSA that accepts exactly palindromes. However, you *can* design an FSA that accepts exactly length-7 palindromes. This might be useful if you *also* wanted your phone number in question 4c to be a palindrome.
Draw such an FSA. To keep it small, limit it to the alphabet $\{0, 1\}$ only. Try to use as few states as possible. (*Hint:* It will still be annoyingly large. What must the state remember?)

(b) [3 points] Write a CFG that accepts exactly palindromes (of *any* length) over alphabet $\{0, 1\}$.

(c) [6 points] Every string in $\{0, 1\}*$ can be divided into a sequence of palindromes. You would like to write a program that partitions an input string in $\{0, 1\}*$ into as *few* palindromes as possible. For example, you want to partition `011000`

into `0110 00` (2 palindromes), not `0 11 000` (3 palindromes) or `0 1 1 0 0 0` (6 palindromes).

You can solve this problem using a probabilistic parser! Write a PCFG, with probabilities, such that the *most probable parse* of a string will reveal its division into *the minimum number of palindromes*.

6. This question is about the *worst-case* asymptotic runtime of parsing algorithms. n is the number of words in the input sentence. V is the number of nonterminals in the grammar. G is the size of the grammar, measured as the total length of all rules, where the length of $A \rightarrow B\ C\ D\ E$ is considered to be 5.

   (a) [3 points] The runtime of Earley's algorithm is the total runtime of all of the SCAN, PREDICT, and ATTACH operations.

   Which type of operation has the *slowest* runtime per operation?
   (*circle one operation, or two operations if neither one dominates the other asymptotically*) SCAN PREDICT ATTACH

   Which type of operation has the *slowest* total runtime when summing over all operations of that type?
   (*circle one operation, or two operations if neither one dominates the other asymptotically*) SCAN PREDICT ATTACH

   (b) [6 points] Earley's original algorithm used entries of the form $(4, A \rightarrow B\ C\ .\ D\ E)$. But in class, we also discussed a modified version that would use entries of the form $(4, A \rightarrow \cdots\ .\ D\ E)$.

   Consider an ATTACH step, which attaches a complete entry (i.e., complete constituent) to its customers. How many complete entries are in the table, altogether, in the worst case?
   (fill the exponents into the boxes; note that 0 is a valid exponent)

   - Under Earley's original algorithm: proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$

- Under the modified algorithm: proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$

(c) [6 points] For a given complete entry that starts at position $n-1$, what is the worst-case runtime of finding all of its customers? Assume Earley's original algorithm in this case.

- By linear search through column $n-1$: proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$
- Using auxiliary data structures such as a hash table to speed up the search for customers: proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$

(d) [3 points] Using your previous answers, how fast can you make the worst-case *total* runtime of all ATTACH operations?

- Under Earley's original algorithm: proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$
- Using the modified algorithm and auxiliary data structures: proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$

(e) [4 points] To use CKY on the same problem, you'd have to binarize the grammar first. The size of the binarized grammar, in the worst case, is proportional to $V^{\square} \cdot G^{\square}$. (Here V and G are measured on the original, unbinarized grammar—the question is how badly G will blow up as a result of binarization.)

(f) [3 points] How fast can you then make the worst-case total runtime of CKY, if you are careful about loop ordering?

- proportional to $n^{\square} \cdot V^{\square} \cdot G^{\square}$

(g) [4 points] Consider a probabilistic or weighted parser. How is w in the modified Earley's algorithm related to $w_1, w_2, \ldots$ in the original Earley's algorithm?

Here w denotes the weight of the modified entry $(4, A \rightarrow \cdots \ . \ D \ E)$. And $w_1, w_2, \ldots$ denote the weights of the corresponding original entries, such as $(4, A \rightarrow B \ C \ . \ D \ E)$ and $(4, A \rightarrow Q \ . \ D \ E)$.

7. You are evaluating a system for identifying dish names within restaurant reviews. The system identifies the names using BIO tagging of the words. Here is one sentence shown with the "gold standard" labels provided by a human, and the system's labels.

gold:	O	O	B	I	O	O	O	O	O	O	O	O	O	B	I	O	
	The	homemade	buffalo	sauce	,	served	with	flair	,	goes	surprisingly	well	on	fried	rice	.	
system:	O	B	I	I	O	O	O	B	O	O	O	O	O	B	I	O	

(a) [7 points] The system's tagging accuracy on this sentence is clearly 13/16. However, the tagging is just a means to an end. You would rather report whether it found the correct dish names. (No partial credit on a dish name: either the system finds it exactly, or it doesn't.)

In the example sentence above, **place brackets around each dish name that the system found.** Then fill in the following:

precision = _____ recall = _____ F_1 = _____

(b) [2 points] Give an example of an "illegal" BIO tag sequence of length 5—one that the system should never be allowed to produce on a 5-word sentence.

(An illegal sequence violates the conventions of BIO tagging. It does not encode any bracketing. So if the system did produce it, you would not even be able to compute precision/recall/F_1 as in the previous question.)

(c) [3 points] Suppose the system uses a weighted finite-state transducer `M` as its tagger. It finds the best tagging of sentence x by computing the best path in `[M .o.` x`].u` (XFST notation), or equivalently, the best path in `Project[Compose[M,`x`],'input']` (Thrax notation).

Unfortunately, sometimes the best path is an illegal tag sequence. Without changing `M` or its weights, how can you modify this system to find the best *legal* sequence? Your answer should somewhere contain a precise characterization of legal sequences.

(d) [2 points] You build your own system for this task, which assigns a BIO tag sequence to sentence x using first-order HMM tagging with posterior decoding, just like homework 6.

Recall that posterior decoding returns the tag sequence with the maximum expected number of correct tags. It makes use of $p(T_i = \texttt{B} \mid x)$, that is, the posterior probability that the tag of word i is `B`.

Give the standard formula that allows you to compute this posterior probabilities from α and β probabilities.

(e) [10 points] Unfortunately, when you implement posterior decoding as in homework 6, it sometimes produces illegal sequences.

A proper posterior decoder will return the legal tag sequence with the maximum expected number of correct tags.

This is a little harder, but that sequence can be found by dynamic programming. Below, write correct pseudocode for that algorithm. Assume that the input sentence is $x = x_1 x_2 \ldots x_n$.

Hint: You have never written this dynamic programming algorithm before, but it is quite similar to other such algorithms in this course. The expected number of correct tags in $t_1 t_2 \ldots t_n$ is $\sum_{i=1}^{n} p(T_i = t_i \mid x)$, so that is what you want to maximize. Assume that you have *already* run the forward-backward algorithm and computed $p(T_i = t_i \mid x)$ for all positions $i = 1, 2, \ldots n$ and all possible values $t_i \in \{\texttt{B}, \texttt{I}, \texttt{O}\}$. Your job now is to stitch those probabilities together into the best possible *legal* sequence.

(f) Another way to build a BIO tagger for this task would be to construct a CRF (conditional random field) that defines a probability distribution $p(y \mid x)$ over BIO tag sequences y given the sentence x.

A CRF is efficient at test time provided that the features are chosen carefully. You decide to design your features such that your CRF tagger can be decoded just as efficiently as in questions 7c or 7e, once the features have been computed.

As usual, each of your CRF features will detect some specific configuration within an (x, y) pair, and will fire on every copy of that configuration.

i. [2 points] Consider the dish name `buffalo sauce`. One clue that this is a dish name is that it immediately follows the particular word `homemade`. To capture this insight, what configuration should you add? (*Warning:* If your answer doesn't explicitly refer to B, I, or O tags, then it is not specific enough.)

ii. [3 points] Another clue that `buffalo sauce` is a dish name is that it ends at the end of an NP (`the homemade buffalo sauce`). To capture this insight, what configuration should you add? Assume that your feature functions have access to a parse of the input sentence. (The parse might be automatically produced and have a few errors, but that's okay: it will merely make the features a little less reliable, so you'll learn lower weights for them.)

8. (a) [2 points] What is the edit distance between `meter` and `metric`? _____

(b) [3 points] Consider the standard FST for edit distance. Don't draw the FST, but draw the minimum-cost path in the FST that aligns `meter` to `metric`. Each arc in your path should show its cost, as well as a label such as `m` : ϵ or `m` : `m`.

(c) Some edits are more likely than others. Suppose we have a training set of pairs $(x_1, y_1), (x_2, y_2), \ldots$. For example, (`meter`, `metric`) or (`leaf`, `leaves`). We'd like to learn which edits are likely.

We could attempt to train new costs for the FST by the following method:

- For each pair (x_i, y_i), find the best path in the FST.
- For each arc a in the FST, let $c(a)$ be the number of times it was traversed in all of these best paths.
- For each arc a in the FST, update its weight to $-\log(c(a)/\sum_a c(a))$.
- Repeat until the weight updates are no longer changing the weights very much.

i. [3 points] The full name of this method is _____

ii. [3 points] In this case, it attempts to maximize _____

iii. [2 points] One reason it might fail to find the maximum is:

iv. [2 points] Another reason it might fail to find the maximum is:

9. This question is about *possible-worlds* semantics. Let's introduce some conventions. To say that Mary loves John at time t in world w, we'll write

$$\text{love}_{w,t}(\text{Mary}, \text{John})$$

Here w and t are merely two extra arguments to the love function, but we write them as subscripts to make it easier to identify them in the notation.

The present-tense sentence `Mary loves John` gets the semantics

$$\lambda w \ \text{love}_{w,\text{now}}(\text{Mary}, \text{John})$$

(where "now" denotes the specific time at which the sentence is spoken). Why does this start with λw? Because the context of this sentence is needed to specify the world w in which Mary supposedly loves John. If I utter it as a ROOT sentence, then I am asserting the above predicate is true of the real world:[1]

$$(\lambda w \text{ love}_{w,\text{now}}(\text{Mary}, \text{John}))(\text{realworld}) = \text{love}_{\text{realworld,now}}(\text{Mary}, \text{John})$$

Whereas if I say `Crazy Boo thinks that [Mary loves John]`, then I am discussing whether the same predicate is true of the world inside Crazy Boo's head.

One more bit of introduction. If you *become* wise at time t, this means that you are wise at time t but not at time $t-1$. Suppose that we represent the semantics of the word `become` as $\lambda pred\ \lambda t\ \lambda subj\ \lambda w\ \left(pred_{w,t}(subj) \wedge \neg pred_{w,t-1}(subj)\right)$

(a) [4 points] What is an appropriate semantics for `Elmo will become president`? Just like `Mary loves John`, above, it should start with λw. Assume that the statement is talking about future times within world w. (*Hint:* Imagine how the semantics of `become`, given above, might combine with other phrases in the sentence. You will have to figure out how to represent future tense.)

(b) [9 points] To obtain this semantics, what semantics should you associate with each of the following constituents? (*Hint:* This is like division.)

- will become president _____
- become president _____
- will _____

[1] Or more precisely, I am asserting that it is true of whatever world is currently under discussion—perhaps I say `Mary loves John` as part of a fairy tale or a hypothetical situation.

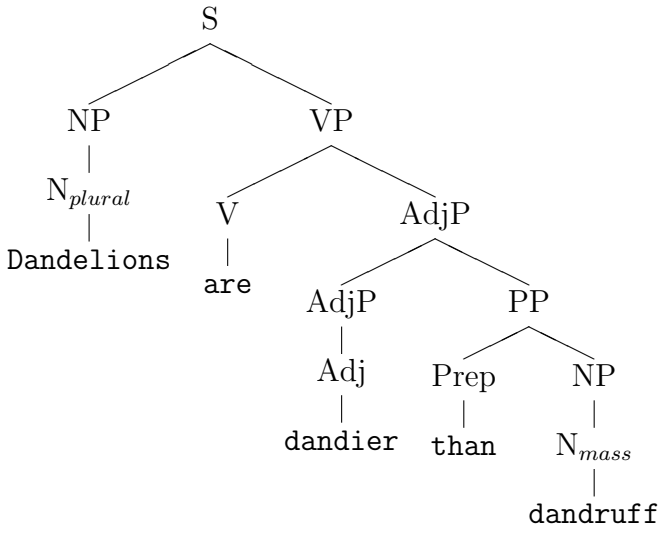
(c) [4 points] Consider the sentence `Grover became president.`[2] What syntactic analysis should we adopt for this sentence so that we can compute its semantics analogously to the above treatment of `Elmo will become president`? (It certainly seems as if the two sentences are analogous: one is past and one is future. The trouble is that the Grover sentence doesn't have a word like `will` to indicate the tense . . . how will you handle that?) Answer by drawing a tree.

(d) [3 points] Consider the sentence `Anyone can become president.` What would a reasonable semantics for this sentence be? Note that the sentence is considering possible worlds w' other than the current world w.)

(e) [3 points] Arguably, `Elmo will become president` is also considering possible worlds other than w: we don't know what the future will hold, so there are many possible future worlds w' that "extend w into the future." How could you have answered question 9a so that it would be more like your answer to question 9d?

[2] Indeed, Grover Cleveland became president twice: he was both the 22nd and the 24th president of the U.S., the only person to serve non-consecutive terms.

10. Here is the best parse of a certain silly sentence, under a PCFG that was trained on a large quantity of general English text.



A constituent of the parse is a nonterminal node (perhaps a preterminal) together with all of its descendants.

Feel free to explain your answers to the following questions.

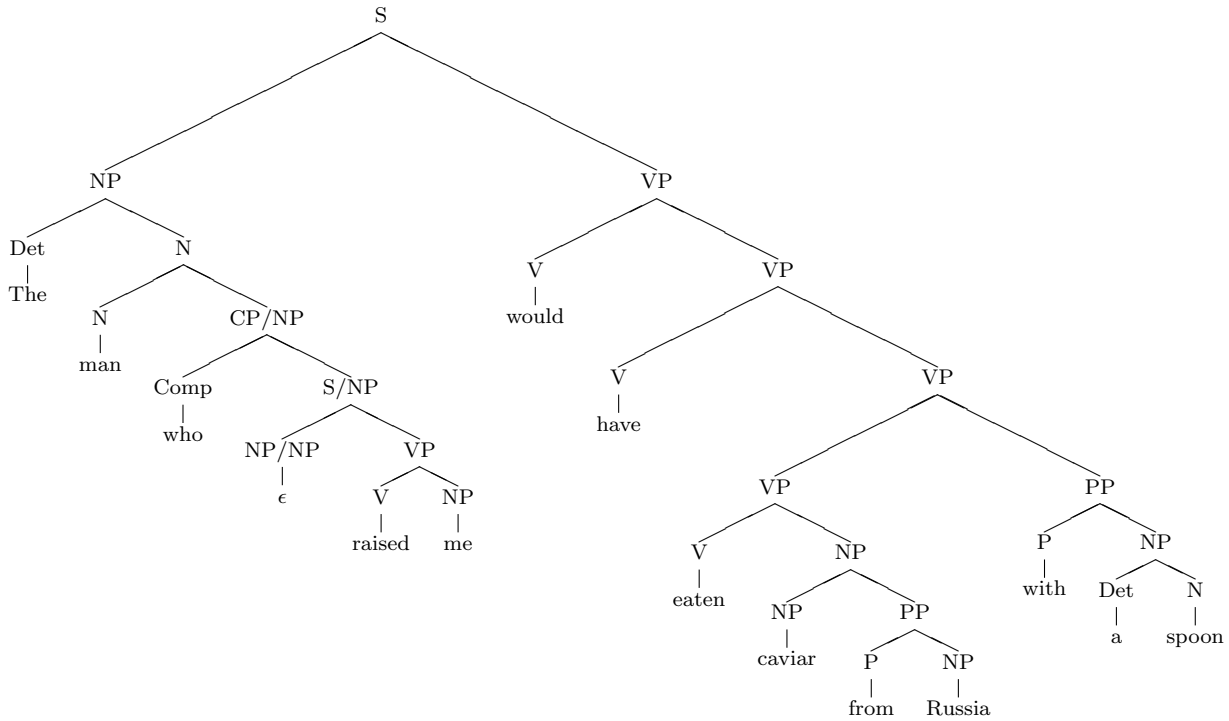
(a) [3 points] Draw a solid circle around the entire constituent that probably has the *smallest inside probability*.

(b) [3 points] Draw a dashed circle around the entire constituent that probably has the *smallest outside probability*.

(c) [3 points] If you raise the probability of the rule AdjP $\rightarrow$ AdjP PP, then the outside probabilities of some constituents will increase. Draw a rectangular box around each entire constituent whose outside probability is *guaranteed* to increase.

(d) [2 points] It is possible that the outside probabilities of some other constituents will increase a little bit, too. Why?

11. (a) [4 points] Next to each node of the following parse, write its head word (if the node dominates more than one word). Assume that `eaten` is the head word of the whole sentence, and that `who` is the head word of the relative clause `who raised me`. Make reasonable assumptions about the head words of other phrases.



(b) [5 points] Now convert this to a dependency parse. That is, in the copy of the sentence below, draw an unlabeled arrow <u>to</u> each word <u>from</u> its parent word. (The sentence is ambiguous, but you should specifically show the dependency parse that corresponds to the phrase-structure parse above.)

`The man who raised me would have eaten caviar from Russia with a spoon`

12. (a) [10 points] Someone is eating apples. Naturally you ask, *Whose apples is she eating?* Professor Boozle corrects you, saying that since a plural subject requires a plural verb, you should ask, **Whose apples are she eating?*

You argue that your version sounds much better in standard English. Prove to Prof. Boozle that your version is actually the grammatical one, by drawing a tree for your sentence in which you show all number features (*num=singular*, *num=plural*). Please also show any tense features (*tense=present, past, progressive, perfect, stem, infinitive, ...*). Feel free to abbreviate the features.

Hint: First try drawing a tree for this version: *She is eating <u>whose</u> apples?* Now draw the requested tree, using slashed nonterminals as necessary.

(b) [5 points] Professor Boozle says, "Now I get it! Thank you for teaching me English grammar. So tell me: **Which barrel are the apples in delicious?*"

That last sentence sounds terrible to you, but Prof. Boozle gives you a taste of your own medicine and draws a tree for it. You concede that Prof. Eisner has not taught you anything that would rule out this tree syntactically. You realize that you also can't think of any semantic reason that the sentence sounds so bad (since the tree seems to express a meaningful question), and conclude that there must be (even) more to linguistics than you can learn in a CS course.

What was Boozle's tree? You don't have to show tense or number features this time, because Boozle didn't. *Hint:* Try to adapt the hint from the previous question—that is, come up with another version of Boozle's question that does sound okay.

Scratch Paper!

Name: _____ SSN: _____

Intro to NLP (600.465) — Final Exam

Johns Hopkins University (Prof. Jason Eisner)
Tuesday 17 December 2002, 2 pm–5 pm
Total points: 180 Total minutes: 180

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

1. Ryan Researcher thinks d is not very important, so he approximates $p(a, b, c, d \mid e, f)$ by $p(a, b, c \mid e, f)$. You should assume that Ryan uses unsmoothed estimates.

   (a) [2 points] What are the *possible* effects of Ryan's approximation on the probability in question? *(circle <u>all</u> that apply)*

   INCREASES PROBABILITY NO CHANGE DECREASES PROBABILITY

   (b) [2 points] Under what conditions is the approximation within 1% of correct? Give your answer in the simplest form possible.

   (c) [2 points] What are the *possible* effects of this approximation on the perplexity of test data? *(circle <u>all</u> that apply)*

   INCREASES PERPLEXITY NO CHANGE DECREASES PERPLEXITY

   (d) [2 points] Is it fair to compare the resulting perplexity to another system's? Briefly say why or why not.

2. [9 points] Which of the following algorithms are greedy algorithms that, because of their greediness, may not get the optimal answer? *(circle <u>all</u> that apply)*

   (a) language ID by comparing n-gram probabilities
   (b) probabilistic CKY parsing
   (c) probabilistic Earley parsing
   (d) transformation-based learning
   (e) Viterbi tagging
   (f) competitive linking
   (g) generating the correct pronunciation with Optimality Theory
   (h) growing a decision tree
   (i) SVD (also known as PCA or Latent Semantic Analysis)

no longer covered

3. [3 points] Given a fixed context-free grammar in Chomsky Normal Form (CNF), how many parses are possible, maximum, for a sentence of n words? *(circle <u>one</u>)*

   (*Note:* "$O(n^2)$" is used here to mean "some unspecified function that grows quadratically with n"; technically, what I mean is $\Theta(n^2)$, but fewer people know that notation.)

   (a) $O(n)$
   (b) $O(n^2)$
   (c) $O(n^3)$
   (d) $O(n^6)$
   (e) exponential on n, i.e., $O(b^n)$ for some constant b

4. [2 points] Given a sentence of fixed length, how many parses are possible, maximum, under a CNF grammar with k nonterminals? *(circle <u>one</u>)*

   (*Note:* $O()$ has same meaning as for the previous question.)

   (a) $O(k)$
   (b) $O(k^2)$
   (c) $O(k^3)$
   (d) $O(k^6)$
   (e) exponential on k, e.g., $O(b^k)$ for some constant b

5. Optical Character Recognition (OCR) is a technology that extracts text from scanned images of the printed page. You wish to build a system for correcting its output. For example, where the OCR device returns cide, which is not an English word, you might correct it to icicle. This is because you have learned from training data that the OCR device often drops an initial i and often confuses cl with d (they look similar!).

   Let x be the string produced by OCR (e.g., cide), and let y be a candidate correction (e.g., icicle).

   (a) [2 points] Suppose you have x as well as 99 candidate corrections, $y_1, y_2, \ldots y_{99}$. You want to choose the best correction. Write the conditional probability you should maximize, and express it using Bayes' Theorem with a summation in the denominator.

   (b) [4 points] Suppose you have available these weighted finite-state machines:
   - `X` = a straight-line FSA accepting only x (with weight 1)
   - `WX` = an FSA that accepts x with weight $p(x)$, where p is a letter n-gram model trained on OCR output
   - `WY` = an FSA that accepts y with weight $p(y)$, where p is a letter n-gram model trained on English output
   - `XY` = an FST that transduces x to y with weight $p(x \mid y)$, where p is a model that evaluates the probability that the OCR device will misread y as x; for example, it considers the probability the OCR device will drop i and replace cl with d

   You want to use some or all of these machines to find the best correction y from the space of *all* possible corrections (not just $y_1, y_2, \ldots y_{99}$). What FSA should you construct (write a regular expression for it in terms of the machines above), and what should you do to that FSA to find y?

(c) [3 points] *Fill in the blanks*: Let $n = 2$, so `WX` is a letter bigram model. Assuming that `WX` assigns positive probability to any ASCII string, then `WX` has _____ states and _____ arcs. (*Note:* ASCII allows 256 different letters, including a special 0 character used to terminate strings.)

(d) [3 points] Same question for $n = 3$: _____ states and _____ arcs.

6. [6 points] "Saccades" are sudden, unconscious eye movements that glance at objects in the environment. Which of the following are true? *(circle all that apply)*

covered in one of the Tuesday sessions (when we had a lecture instead of recitation)

(a) Humans make 7–8 saccades per second, on average.

(b) Humans do not continuously track objects with their eyes; this is an optical illusion formed by a series of rapid saccades.

(c) Eye-tracking experiments cannot discern a syntax-only "first pass" of parsing that is later revised using semantic information.

(d) Saccades to a named object occur (at a rate greater than chance) before the object's name or description is completely spoken.

(e) Eye-tracking devices require the subject's head to be immobilized.

(f) Before eye-tracking was developed, the dominant paradigm for studying human sentence processing was electroencephalography (EEG), in which electrodes are attached to the scalp to measure brain electrical activity.

(g) none of the above

7. [6 points] You are clustering 6 two-dimensional data points into 2 clusters, using EM clustering. You compute the following data likelihoods, based on an assumption that each cluster's true distribution has variance 3, and your current guess that the centroids of the "A" and "B" clusters are at (0,0) and (5,5) respectively:

no longer covered but see hint!

data point		data likelihoods	
x	y	$p(point \mid A)$	$p(point \mid B)$
1	1	0.165	0.001
1	2	0.100	0.004
1	3	0.044	0.008
2	4	0.008	0.044
3	4	0.004	0.100
4	4	0.001	0.165

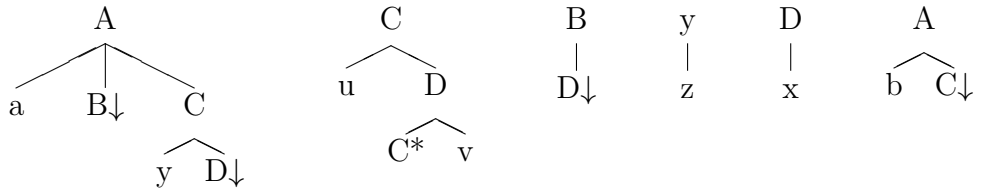
When you recompute the cluster centroids, what is the x coordinate of the new centroid for the "A" cluster? (Write this as an arithmetic expression in terms of the numbers above; you don't have to actually compute its value.)

(*Hint:* If you don't remember how to do EM clustering, it's similar to the forward-backward algorithm, except there are no transition probabilities to worry about. Basically you are tagging each point with "A" or "B" independently of the others. So imagine that you have to tag 6 sentences of one word each.)

8. [5 points] Here are some elementary trees of a Tree-Adjoining Grammar (TAG). Assemble some of them into a complete tree that is permitted by this grammar. You must use at least the first two elementary trees.

no longer covered

(*Note:* Capital letters are nonterminals, lowercase letters are terminals. I have also observed the alternative TAG convention of marking nonterminal leaves with ↓ or *.)



9. [4 points] Give an example of each:

   (a) a single-morpheme word with more than one sense: _____

   (b) a morphologically ambiguous word containing at least 2 morphemes: _____

   (c) a syntactically ambiguous sentence that contains no prepositional phrases

   (d) a syntactically unambiguous sentence with ambiguous semantics; in this case, explain the two meanings. You may not use the type of ambiguity illustrated in question 22. (*Hint:* quantifier order.)

10. [8 points] A "simple regexp" is a regular expression that only uses the concatenation, union (`+`), and closure (`*`) operators. Write a CFG that generates simple regexps, with semantic attachments that compute the corresponding the finite-state machine. For example, `a+b(c+d)*` should get the semantics `plus(a,concat(b,star(plus(c,d))))`.

    Note: As in the example, `*` takes precedence over concatenation, which takes precedence over `+`. Don't forget to deal with parentheses, as in the example!

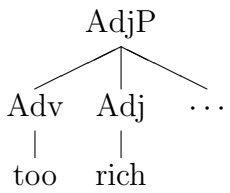
    Note: Use the conventions of assignment 3, and don't forget a `START` symbol.

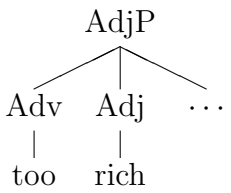
    Hint: No λ symbols are necessary.

    Here are a few rules to get you started:

```
Factor[sem=1] → a
Factor[sem=1] → b
Factor[sem=1] → c
Factor[sem=1] → d
Term[sem=1] → Factor
```

11. [7 points] Supporters of Mike Bloomberg's successful mayoral campaign described him as "too rich to steal." They could also have described him as "too rich to bribe." Although these two adjective phrases (`AdjP`) are both headed by "rich" and make more or less the same political argument, there is a subtle syntactic difference between them. Draw both `AdjP` trees in a way that explains the difference. I've gotten you started—just fill in the "$\cdots$". (*Hint*: an empty or silent constituent has terminal symbol e.)





12. [5 points] The forward-backward algorithm has which of the following problems? *(circle <u>all</u> that apply)*

    (a) The algorithm can never find the true probabilities if probabilities that are truly nonzero are initialized as 0.

    (b) The algorithm can never find the true probabilities if transition probabilities that are unequal are initialized as equal.

    (c) The algorithm can never find the true probabilities if there are local maxima.

    (d) The algorithm may decrease perplexity on training data, but increase error rate on test data (measured against human annotations).

    (e) The algorithm may decrease error rate on test data (measured against human annotations), but increase perplexity on training data.

    (f) none of the above

13. 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?

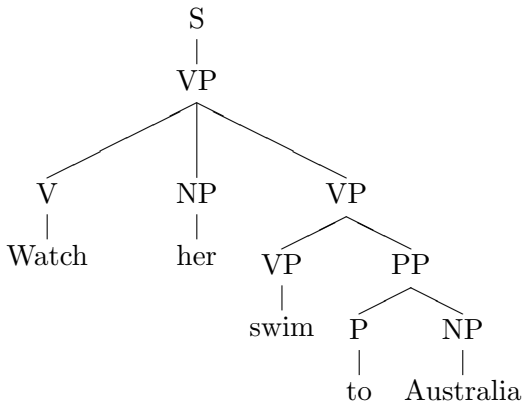
    (c) [4 points] In your analogy from (a), what kind of natural language objects correspond to:

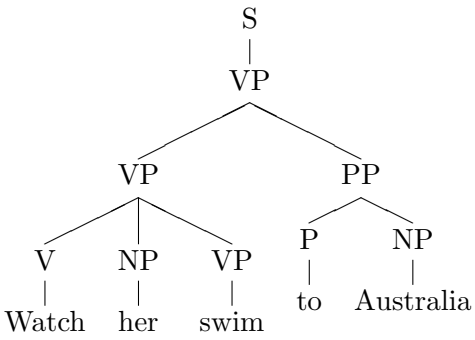
    - states? _____
    - days? _____
    - dollars? _____

    (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.

14. [5 points] The following two trees have different PP attachments, but since they use the same set of rules, they have exactly the same probability under any probabilistic context-free grammar (PCFG). In other words, the ratio of their probabilities is 1, so there is no way to see that the first is more probable. This is a serious problem with PCFGs!





The usual solution is to enrich the nonterminals with head words, e.g., `V[head=watch]`. Then what is the ratio of the two trees' probabilities? (In your answer, you should cancel out the factors they have in common.)

(*Note:* You may want to indicate the headwords on the trees above, so that if you chose them strangely we can at least figure out what your answer was supposed to mean.)

15. Whether the parentheses in a string are properly balanced (i.e., whether they match and nest correctly) cannot be checked by any finite-state automaton . Roughly speaking, the problem is that finite-state devices can't count arbitrarily high; they can't even check that there are equally many left and right parentheses.

    However, you can build an *approximate* FSA for this task. Your FSA will be able to count up to 5 levels of nested parentheses.

    (a) [5 points] An "innermost" pair of parentheses is `(` and `)` with no other parentheses between them. Define a finite-state transducer T that removes all innermost pairs of parentheses from its input. You should *not* assume that the input is properly balanced.

    For example, applying T to `(a))b(c(de)f()gh)))i(` should remove the parentheses around `(a)`, `(de)`, and `()`, giving `a)b(cdefgh)))i(`. Applying T again to this should give `a)bcdefgh))i(`, which will be unchanged by any further applications of T.

    Your answer should be an expression in XFST notation. You may assume the following definitions (and you are free to define other intermediate quantities):

    Thax also ok

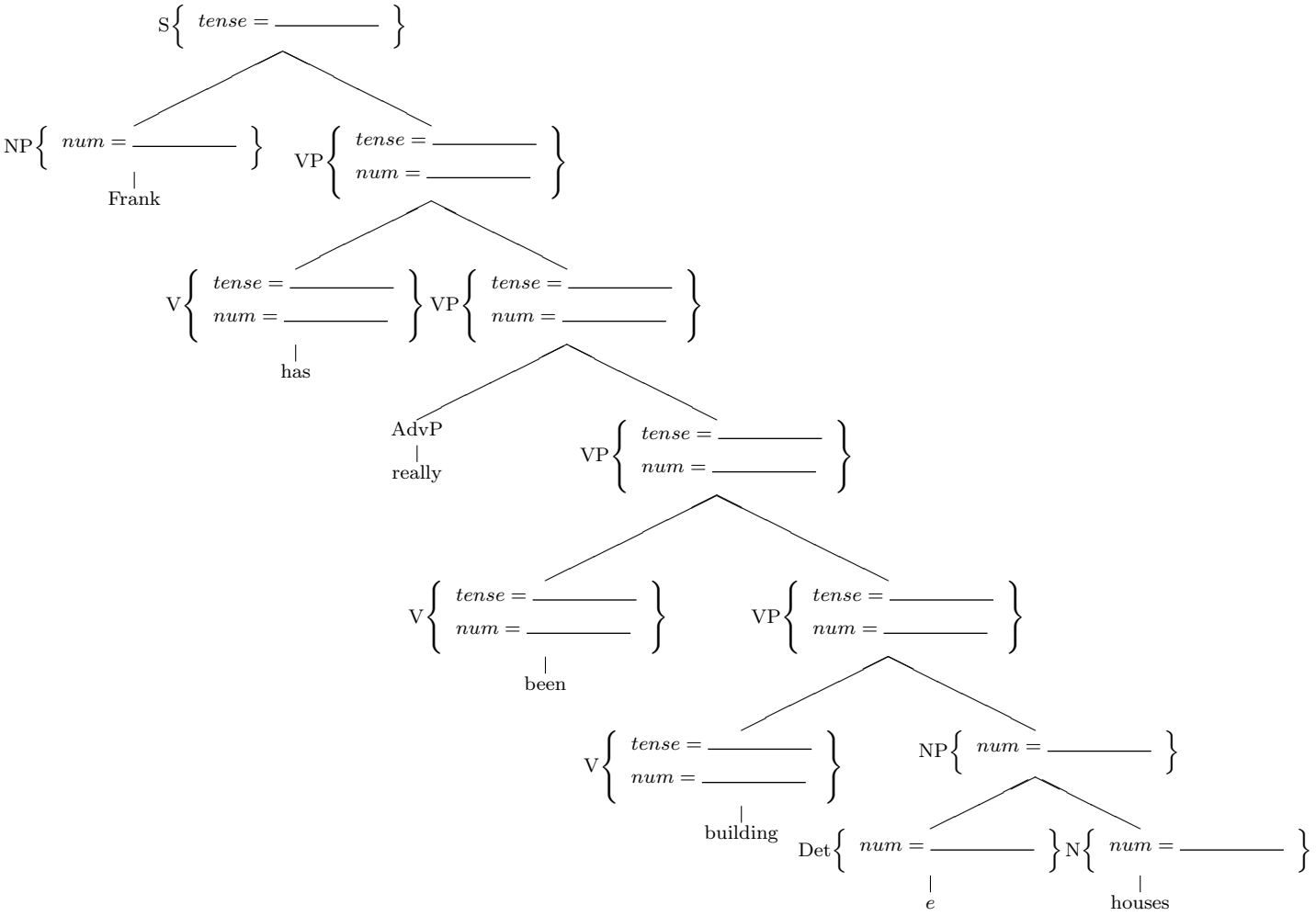
    ```
    define L %( ;
    define R %) ;
    define C ?-L-R ;
    ```

    Hint that was not on the actual exam: Replacing `L C* R` with `C*` doesn't work, since the replacement string doesn't have to have anything to do with the search string (it is chosen nondeterministically from `C*`). On the other hand, replacing `C*` with `...` doesn't change anything. You'll have to work harder than that!

    (b) [4 points] Using T, define an acceptor A (not a transducer!) such that A accepts only strings whose parentheses are properly balanced *and* are nested to at most 5 levels deep. Your definition of A should be in XFST notation, in terms of T, and it should work for any correct definition of T.

    For example, A should accept `f(3*(x+4)^(1/g(5)))`. But it should reject `(a))b(c(de)f()gh)))i(` because it is unbalanced, and it should reject `((((((x))))))` because the parentheses are nested 6 levels deep.

16. [7 points] Most of you had trouble with this on the midterm, so here's another chance to show that you really do understand features for auxiliary verbs. Fill in the blanks for tense and number features.



17. You are trying to classify web pages as to whether they are about computational linguistics. The term `NLP` is a reasonably good indicator (but it also appears on many pages about the pseudo-science of "neurolinguistic programming"). Another good indicator is the term `disambiguation`.

Here are some approximate counts for web pages in the Google Directory:

	has `NLP`?		has `disambiguation`?		
	Yes	No	Yes	No	Total
docs about comp. ling.	104	206	40	270	310
all other docs	783	≈7 million	35	≈7 million	≈7 million
Total	887	≈7 million	75	≈7 million	≈7 million

Note: In the following questions, write your answer as an arithmetic expression in terms of the numbers above; you don't have to compute its value. Write "≈7 million" as "7M."

(a) [2 points] A classifier trained only on the "has `NLP`" feature would have recall of _____ and precision of _____.

(b) [2 points] A classifier trained only on the "has `disambiguation`" feature would have recall of _____ and precision of _____.

not really covered this year

(c) [6 points] You are given a document ~~with~~ "has `NLP`"=No and "has `disambiguation`"=Yes. Using a Naive Bayes classifier on these two features, trained on the above data without smoothing, what is the probability that the document is about computational linguistics? Is this probability > 0.5 (you should be able to figure this out with a very small computation)?

(*Hint*: Start by writing down the formula in the abstract; then write down the probabilities you will need, e.g., p(CL) and p(Other); finally, substitute the probabilities into the formula.)

18. [6 points] In the situation of the previous problem, let x be the document you are trying to classify, and let c be a candidate classification. But now you are using a maximum entropy (log linear) model. Here are the weights of the relevant features of the pair (x, c):

0	$c =$CL	14	$c =$Other
-2	$c =$CL and x has `NLP`	-9	$c =$Other and x has `NLP`
-1	$c =$CL and x does not have `NLP`	0	$c =$Other and x does not have `NLP`
-3	$c =$CL and x has `disambiguation`	-8	$c =$Other and x has `disambiguation`
0	$c =$CL and x does not have `disambiguation`	0	$c =$Other and x does not have `disambiguation`

As in the previous problem, x does not have `NLP` but x does have `disambiguation`. Find $p(x, c = \text{CL})$ and $p(x, c = \text{other})$ (just write the necessary normalizing term as $\frac{1}{Z}$ without trying to compute it). From these, find $p(c = \text{CL} \mid x)$.

19. [10 points] In 3–4 sentences, describe any reasonable method of distinguishing between the two senses of "star" (movie star and sky star). Be sure to indicate how the method is trained, and what training data it requires, if any.

20. [5 points] You can compute word bigram probabilities either forward,

$$p(w_1 \mid \text{START}) \cdot p(w_2 \mid w_1) \cdots p(w_n \mid w_{n-1}) \cdot p(\text{STOP} \mid w_n)$$

or backward,

$$p(w_n \mid \text{STOP}) \cdot p(w_{n-1} \mid w_n) \cdots p(w_1 \mid w_2) \cdot p(\text{START} \mid w_1)$$

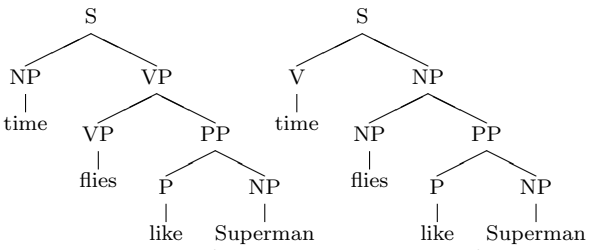
The two computations give the same result under which kinds of smoothing? *(circle <u>all</u> that apply)*

(a) no smoothing
(b) add-λ smoothing
(c) add-λ smoothing with backoff
(d) Katz backoff
(e) one-count smoothing as in assignment 5
(f) none of the above

not covered this year

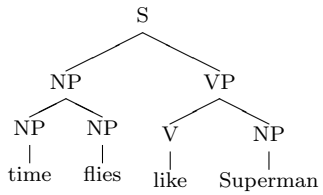
21. Here is a probabilistic CFG in Chomsky Normal Form, together with the 5 parses it allows for *time flies like Superman* and their probabilities. The numbers in the grammar are probabilities, not log-probabilities: e.g., $p(\text{S} \rightarrow \text{NP VP} \mid \text{S}) = 0.6$. Not all terminal rules are shown.

0.6	S	→ NP VP	0.01	NP	→ Superman
0.2	S	→ S PP	0.02	NP	→ flies
0.2	S	→ V NP	0.03	NP	→ time
0.5	VP	→ V NP	0.04	VP	→ flies
0.3	VP	→ VP PP	0.05	V	→ time
0.4	NP	→ NP PP	0.06	V	→ like
0.1	NP	→ NP NP	0.07	P	→ like
1.0	PP	→ P NP			

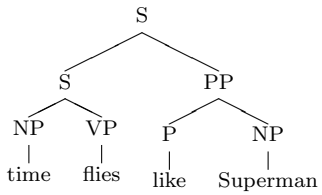


$p \approx 15 \times 10^{-8}$

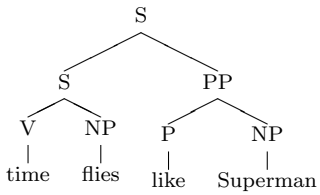
$p \approx 6 \times 10^{-8}$



$p \approx 1 \times 10^{-8}$



$p \approx 10 \times 10^{-8}$



$p \approx 3 \times 10^{-8}$

You are given *time flies like Superman* as raw (unparsed) data during EM training.

(a) [2 points] The total probability of all five trees is $\approx 35 \times 10^{-8}$. What does this number represent?

(b) [2 points] Using the approximate parse probabilities shown above, find p(there is an S from 0 to 2 | *time flies like Superman*).

(c) [2 points] Using the rule probabilities, what is $\beta_S(0, 2)$? (Your answer should be an arithmetic expression in terms of the numbers above.)

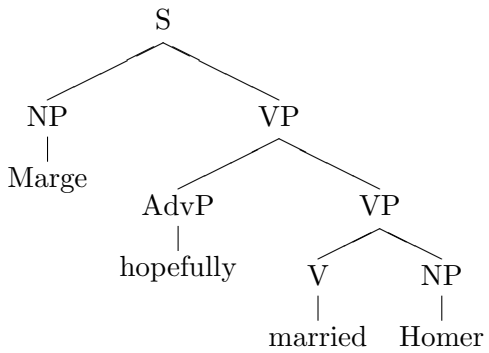
(d) [2 points] Using the rule probabilities, what is $\alpha_S(0, 2)$? (Your answer should be an arithmetic expression in terms of the numbers above.)

(e) [2 points] What is the relationship among the values from the previous parts of this question? (If you want to check your answer, you could compute the actual values for 21c–21d, perhaps rounded to the nearest power of 10.)

(f) [2 points] In general, is it faster to compute p(there is an S from 0 to 2 | *time flies like Superman*) by using your method from 21b (i.e., by enumerating the parses), or by computing and combining α and β probabilities? Why??

22. The following sentences each have one parse (the parse of the first sentence is shown):

Marge hopefully married Homer
Marge certainly married Homer
Marge obviously married Homer



But each of these special adverbs has two possible meanings, giving rise to semantic ambiguity without syntactic ambiguity. In this problem, you will consider the first sentence only. *hopefully* can function as a "sentential adverb" (like *probably*) that expresses the speaker's attitude toward the event:

$\mathsf{hope}(\mathsf{Speaker}, \exists e\ \mathsf{past}(e) \wedge \mathsf{act}(e,\mathsf{marriage}) \wedge \mathsf{marrier}(e,\mathsf{Marge}) \wedge \mathsf{victim}(e,\mathsf{Homer}))$
("I hope that Marge married Homer"—i.e., that their kids are legitimate)

But *hopefully* can also function as an ordinary "adverb of manner" (like *quickly*) that describes Marge's behavior:

$\exists e\ \mathsf{past}(e) \wedge \mathsf{act}(e,\mathsf{marriage}) \wedge \mathsf{marrier}(e,\mathsf{Marge}) \wedge \mathsf{victim}(e,\mathsf{Homer}) \wedge \mathsf{manner}(e,\mathsf{hopeful})$
("Marge married Homer in a hopeful manner"—i.e., hoping he'd improve with age)

(a) [3 points] To obtain the desired semantics for the whole sentence, what semantics do you need for the VP, *married Homer*? (The VP is unambiguous so it should mean the same thing in both versions of the sentence. *Hint:* You might also think about the simpler sentence *Marge married Homer*.)

You may assume the following obvious semantic attachments:

S[sem=2(1)] → NP[sem=1] VP[sem=2]
VP[sem=1(2)] → V[sem=1] NP[sem=2]
VP[sem=1(2)] → AdvP[sem=1] VP[sem=2]
N[sem=Marge] → Marge
N[sem=Homer] → Homer

(b) [3 points] Then what semantics do you need for the verb *married*?

V[sem=_____] → married

(c) [3 points] Then what is the appropriate semantics for the adverb *hopefully* when used as a sentential adverb? (*Hint:* First think about the semantics you want for the two VPs, then how to get from one to the other. Show your work.)

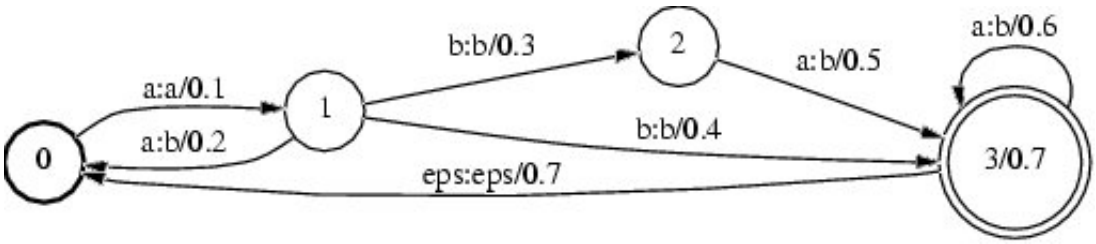
AdvP[sem=_____] → hopefully

(d) [3 points] Similarly, you'd like to find the appropriate semantics for the adverb *hopefully* when used as an adverb of manner. Why *doesn't* the following work?

AdvP[sem=$\lambda f\ (f \wedge \mathsf{manner}(e,\mathsf{hopeful}))$] → hopefully

(e) **[6 *extra credit* points]** Fixing the problem in question 22d actually requires other changes to the grammar. Sketch an approach to fixing it.

23. (a) [2 points] If T is the following finite-state transducer, draw the finite-state transducer for T^*. You can draw it separately or just modify the printed drawing of T.



(b) [2 points] Let U be a finite-state automaton for a^*b. Draw it. Number the states.

(c) [5 points] Draw the finite-state transducer T `.o.` U. The state numbers should be ordered pairs of the form (state in T, state in U).

24. [4 points] Which of the following approaches would be a reasonable approach to pruning a probabilistic Earley's parser? *(circle <u>all</u> that apply)*

    (a) only consider constituents with probability $> 10^{-200}$

    (b) during the SCAN step, don't use the rule $A \rightarrow a$ unless either A or a has probability over a threshold

    (c) during the PREDICT step, increase the probability of the new entry (e.g., VP $\rightarrow$. V NP) if the old entry that called for it (e.g., S $\rightarrow$ NP . VP) had high probability

    (d) none of the above

25. [3 points] Which of the following languages has a click phoneme in its name? *(circle <u>one</u>)*

    (a) Mandarin

    (b) Xhosa

    (c) Tagalog

    (d) English

    (e) Manx

Did you write your name on the first page???