Midterm Exam

COMS W4705: Natural Language Processing

October 21, 2019

Directions

This exam is closed book and closed notes. It consists of three parts. Each part is labeled with the amount of time you should expect to spend on it. If you are spending too much time, skip it and go on to the next section, coming back if you have time.

The first part is multiple choice. The second part is short answer. The third part is problem solving.

Important: Answer Part I by circling answers on the test itself. Answer Part II and Part III questions in the space on the test following the question. Turn in all test sheets at the end of the exam. If you need extra paper, write CONTINUED at the end of the space and put the problem number in bold on top of the extra piece of paper.

Name: _____ UNI: _____

UNI of the person to your left:

UNI of the person to your right:

1 Multiple choice (30 points total)

Recommended time: 20 minutes.

1. (3 points) Which assumption(s) are made by a multinomial Naïve Bayes classifier?

A. The classification problem is binary (i.e., the labels are $\in \{0, 1\}$).

B. The words in a document are independent of each other given the class of the document.

C. The probability of each word appearing in the document depends only on the word that appeared right before it.

D. Documents are represented as vectors of word counts.

Solution:

B. Multinomial Naïve Bayes can be used for multiclass classification; answer C is the first-order Markov assumption and contradicts B; and as covered in the textbook, documents can be equally well represented as sequences of words (so while it is common to use a bag-of-words representation, it is not an assumption specific to the model).

2. (3 points) Given the training data below, estimate the probability of the sequence "I like meatballs." Assume you are using a bigram language model and that punctuation DOES NOT count as a unigram.

Begin training data:

```
I like spaghetti and meatballs.  I like meatballs made of beef.  Yum!
So hungry!
```

End training data.

A. 1/12

B. 1/6

C. 1/8

D. 1/4

Solution:

The correct answer, $\frac{1}{16}$, was not reflected in the answer choices.

$$P(\text{I}, \text{like}, \text{meatballs}) = P(\text{I} \mid \text{<start>}) * P(\text{like} \mid \text{I}) * P(\text{meatballs} \mid \text{like}) * P(\text{<end>} \mid \text{meatballs})$$
$$= \frac{1}{2} * 1 * \frac{1}{2} * \frac{1}{4}$$
$$= \frac{1}{16}$$

Our n-gram counts are...
<start>, I: 2
<start>, Yum: 1
<start>, So: 1
$\implies P(\text{I} \mid \text{<start>}) = \frac{2}{4}$

I, like: 2
$\implies P(\text{like} \mid \text{I}) = \frac{2}{2} = 1$

like, meatballs: 1
like, spaghetti: 1
$\implies P(\text{meatballs} \mid \text{like}) = \frac{1}{2}$

meatballs, <stop>: 1
beef, <stop>: 1
Yum, <stop>: 1
hungry, <stop>: 1
$\implies P(\text{<stop>} \mid \text{meatballs}) = \frac{1}{4}$

3. (3 points) A valid dependency tree is projective if

A. None of the arrows in the tree cross any other arrow.

B. No more than one arrow in the tree crosses other arrows.

C. Every node has only one parent.

D. None of the above.

Solution:

A.

4. (3 points) Which of the following are **true** about perplexity? (Select all that are true.)

A. Perplexity is a measure of how well a grammar or language model models a corpus.

B. A high perplexity means the model is better.

C. A corpus of simple, repetitive sentences, like a children's book, would have a **higher** perplexity than a corpus of complex, varied sentences, like a complicated novel.

D. Perplexities are always greater than 1.

Solution:

A and D.

Quoting Jurafsky and Martin: "The perplexity (sometimes called PP for short) of a language model on a test set is the inverse probability of the test set, normalized by the number of words." Thus it is a measure of how well a grammar or language model models a corpus.

Lower perplexities correspond to higher likelihoods – it is better to be less perplexed.

In the limit of a perfect language model, the perplexity is 1. In practice, language models tend to give perplexities in the range between 1 and V (the vocabulary size.)

Perplexity can also be thought of as the "weighted average branching factor of a language". "The branching factor of a language is the number of possible next words that can follow any word." Thus a more complex corpus will have a higher perplexity (i.e. a lower likelihood) than a simple one.

5. (3 points) The Distributional Hypothesis states... (select only one)

A. That we can distribute predictive power across all words in a sentence.

B. That the frequency of words is distributed according to an inverse power law.

C. That words that occur in the same contexts tend to have similar meanings.

D. That the probability of a word occurring can be accurately modeled with only the n previous words.

Solution:

C.

The distributional hypothesis states that the meaning of words are distributed across the contexts in which they appear. The famous quote is, "You can tell a word by the company it keeps."

6. (3 points) Suppose that we have a vocabulary size of V, and a corpus C in which m unique bigrams are observed, where $m \leq V^2$. You create a bigram language model using the bigram counts from this corpus. However, you have some unobserved bigrams and choose to smooth your language model with add-one smoothing, where you add 1 to the count of every bigram (and renormalize your probabilities appropriately). How much probability mass is removed from observed bigrams to be given to unobserved bigrams (total)?

A. $\sum_{(w_i,w_j)\in C} \frac{count(w_i,w_j)+1}{count(w_i)+V}$, for the m bigrams you originally observed

B. $\sum_{(w_i,w_j)\notin C} \frac{count(w_i,w_j)}{count(w_i)}$, for the V^2-m bigrams you did not originally observe

C. $\sum_{(w_i,w_j)\in C} \frac{count(w_i)+1}{count(w_i)+V}$, for the m bigrams you originally observed

D. $\sum_{(w_i,w_j)\notin C} \frac{1}{count(w_i)+V}$, for the V^2-m bigrams you did not originally observe

Solution:

D, which is the total smoothed probability mass of all the unobserved bigrams (remember that $count(w_i, w_j) = 0$ for any bigram we did not originally observe). Answer A is the new probability mass of the bigrams we **did** observe. Answer B is the probability mass of the unobserved bigrams without smoothing (that is, it is actually 0 because $count(w_i, w_j) = 0$ for any bigram we did not originally observe). In Answer C, we never include unigram counts in the numerator of a bigram language model so this is obviously incorrect.

Note that answer B was mistakenly written with just (w_i, w_j) in the numerator (no "count") for the exam, and we announced a clarification.

7. (3 points) Which of the following statements is **not** true of a discriminative classifier?

A. It learns boundaries between classes.

B. It is probabilistic.

C. It infers outputs based on inputs.

D. It learns weights for features.

Solution:

B. Discriminative approaches are not probabalistic while generative approaches are. For example, Naive Bayes is probabilistic.

8. (3 points) In English, sentences generally follow subject-verb-object word order, where the verb and object form a constituent verb phrase (predicate). This is called "subject-predicate structure". Recall in an English CFG some examples of subject-predicate structure are expressed by the following rules:

```
S -> NP VP (John saw the dog)
S -> Aux NP VP (Did John see the dog?)
S -> Conj VP NP aux (And see the dog, John did!)
VP -> V NP. (see the dog)
```

In which of the following languages would it be impossible for a CFG to express subject-predicate structures?

A. A language with object-verb-subject order, just as Hixkaryana:
"toto yonoye kamara" (lit. "person ate jaguar" — "the jaguar ate the man").

B. A language with subject-object-verb order, such as Turkish:
"Mustafa eşekleri gördü" (lit. "Mustafa donkeys saw" — "Mustafa saw the donkeys").

C. A language with verb-subject-object order, such as Welsh:
"welodd Mair ddraig" (lit. "saw Mary dragon" — "Mary saw a dragon").

D. A language with verb-object-subject order, such as Malagasy:
"Namùnji àzi àhu" (lit. "helped out him I" — "I helped him out").

Solution:

C. To express subject-predicate structure, we must be able to form a predicate: a constituent consisting of the verb and object and have the subject separate. In a CFG, parts of a constituent must be next to each other in the sentence. In option C, the subject is between the verb and the object, and so in order to form a constituent consisting of the verb and object, we would need to include the subject, which is not subject-predicate structure. In all the other options, a verb-object or object-verb constituent can be made.

9. (3 points) In skip-gram with negative sampling, why do we use negative samples rather than simply trying to maximize $\sigma(w \cdot c)$ for (w, c) pairs in our corpus?

A. If we don't have any negative samples, we never learn which word and context vectors should be far apart.

B. If we use $\sigma(w \cdot c)$, we will always converge to the trivial solution where all word vectors are diametrically opposite to all context vectors.

C. Using negative samples increases numerical stability.

D. If we use $\sigma(w \cdot c)$, probabilities will not sum to 1.

Solution:

A. In other words, if we don't have any negative samples, we will always converge to the trivial solution where *all word vectors and context vectors are the same.* Without negative samples, we are just freely maximizing the dot product between each word-context pair that occurs in our corpus, which of course happens in the degenerate case where all vectors are pointing in the same direction and are as long as possible.

10. (3 points) The nth-order version of the Markov assumption states that the probability of occurrence of a word at any given point in time is only dependent on the n words that occurred immediately before it. This is a very strong assumption for low n, but grows more realistic as our history length n increases. Why, then, do we typically choose low values of n, in the 1-3 range? (Select all that apply.)

A. In practice, most in-sentence dependencies do not stretch further than three words back in time.

B. It is easier to model longer sequences of words with more powerful models, such as recurrent neural networks.

C. For efficiency's sake - computation is impractical for large n.

D. Increasing n reduces the number of n-grams we can observe in a sentence, as each n-gram is longer; therefore we are actually losing information when we use higher n.

Solution:

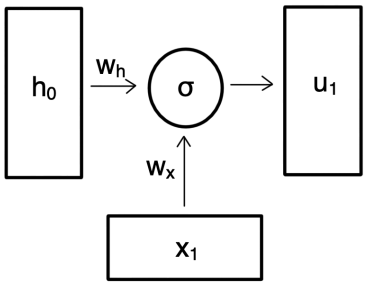
C is true. Answering both B and C is also acceptable. n-gram models deal with quantities of n-grams on the order of $|V|^n$, meaning that as n increases past 4 or 5, we encounter the dual problems of intractability and sparsity of data (in order to collect data with the same frequency for longer n-grams we need an exponentially larger corpus).

2 Short answer (32 points total)

Provide at most 2 or 3 sentences for four out of the following six questions. Each question is worth 8 points.

NOTE: Choose FOUR. If you answer more than four, you will be graded on the first four answers you provide.

Recommended time: 25 minutes.



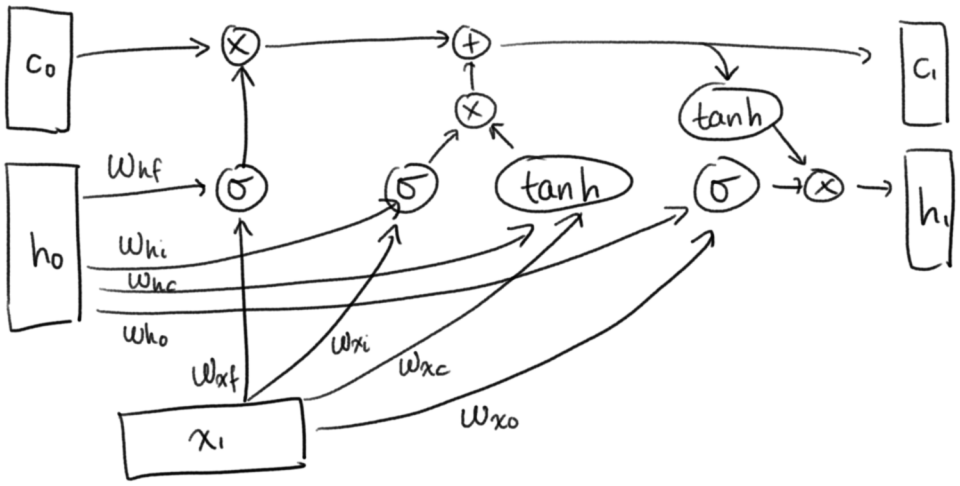
1. (8 points) A diagram of a basic RNN is shown above. To augment it to an LSTM, what would you add and why? Show the augmented diagram of the basic LSTM.

Solution:

An LSTM is intended to improve upon the problem of vanishing gradients, or, more generally, information loss over long sequences, in a vanilla RNN. In addition to the regular hidden state, the LSTM includes a **memory cell** c_0 which is allowed to retain information from arbitrarily far back in the sequence and is not necessarily dominated by the most recent states, as in an RNN. To achieve this, the LSTM also includes three gates which control the flow of information through the memory cell and hidden state:

- An **input gate** i_0, which controls how much of the current input gets passed into the network;
- A **forget gate** f_0, which controls what information stays in the memory cell; and
- An **output gate** o_0, which controls how much of the current timestep gets passed along to the next one.

The gates act as sort of boolean masks to retain and drop information from their associated elements (input, memory cell, and output).



2. (8 points) Consider the following basic neural network.

$$f_1 : \mathbb{R}^3 \to \mathbb{R} \qquad f_1(x) = w \cdot x_1 + b \cdot x_3$$
$$f_2 : \mathbb{R} \to \mathbb{R} \qquad f_2(x) = e^{2x}$$
$$g : \mathbb{R}^3 \to \mathbb{R} \qquad g(x) = f_2(f_1(x))$$

$$L(D) = -\sum_{x^{(i)} \in D} \log g(x^{(i)}).$$

Suppose that we have data $D = \{x^{(1)} = [x_1, x_2, x_3]\} = \{x^{(1)} = [1, 2, -1]\}$ and $w = 2, b = 1$. Assume also that the learning rate $\eta = 0.1$. After one pass through D of minibatch stochastic gradient descent with batch size of 1, what are w and b?

Solution:

$$\frac{\partial L(D)}{\partial w} = \frac{\partial L(D)}{\partial f_2(f_1(x^{(1)}))} \frac{\partial f_2(f_1(x^{(1)}))}{\partial f_1(x^{(1)})} \frac{\partial f_1(x^{(1)})}{\partial w} \quad \text{Chain Rule}$$

$$\frac{\partial L(D)}{\partial b} = \frac{\partial L(D)}{\partial f_2(f_1(x^{(1)}))} \frac{\partial f_2(f_1(x^{(1)}))}{\partial f_1(x^{(1)})} \frac{\partial f_1(x^{(1)})}{\partial b} \quad \text{Chain Rule}$$

$$g(D) = f_2(f_1(x^{(1)})) = e^{2f_1(x^{(1)})} = e^2$$

$$\frac{\partial L(D)}{\partial f_2(f_1(x^{(1)}))} = \frac{-1}{g(x^{(1)})} = \frac{-1}{f_2(f_1(x^{(1)}))} = \frac{-1}{e^2} \qquad \frac{\partial f_2(f_1(x^{(1)}))}{\partial f_1(x^{(1)})} = 2e^{f_1(x^{(1)})} = 2e^2$$

$$\frac{\partial f_1(x^{(1)})}{\partial w} = x_1^{(1)} = 1 \qquad \frac{\partial f_1(x^{(1)})}{\partial b} = x_3^{(1)} = -1$$

$$\frac{\partial L(D)}{\partial w} = \left(\frac{-1}{e^2}\right)(2e^2)(1) = -2 \qquad \frac{\partial L(D)}{\partial b} = \left(\frac{-1}{e^2}\right)(2e^2)(-1) = 2$$

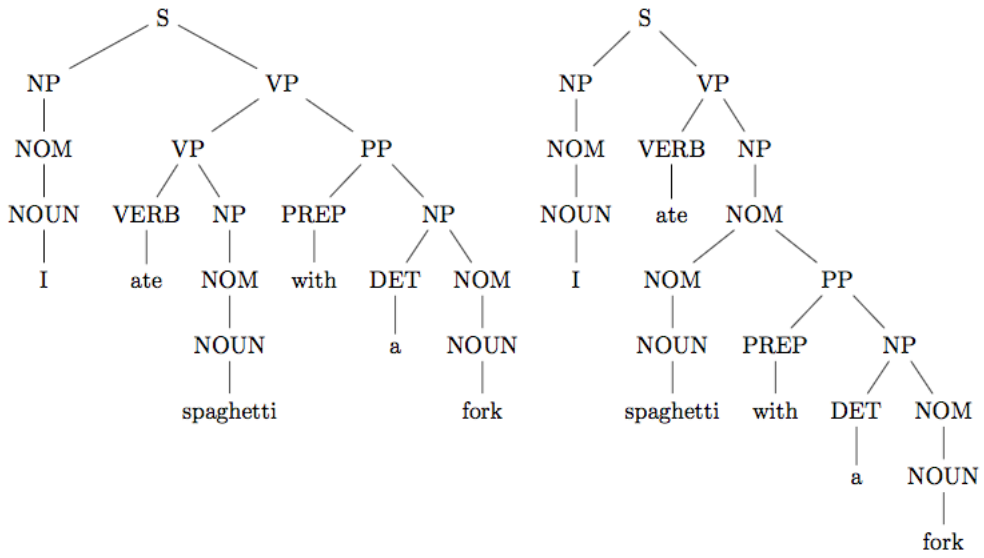
$$w \leftarrow w - \eta \frac{\partial L(D)}{\partial w} = 2 - 0.1(-2) = 2.2 \qquad b - \eta \frac{\partial L(D)}{\partial b} = 1 - 0.1(2) = 0.8$$

3. (8 points) Consider the context free grammar below.

```
S-> NP VP                    NOUN -> I
VP -> VP PP                  NOUN -> spaghetti
VP -> VERB NP                NOUN -> fork
VP -> VERB                   NOUN -> night
NP -> DET NOM                VERB -> ate
NP -> NOM                    PREP -> with
NOM -> NOM PP                PREP -> at
NOM -> NOUN                  DET -> a
PP -> PREP NP
```

(a) Show two different parses that would be derived for "I ate spaghetti with a fork."

Solution:



(b) Augment the grammar to include one or more adjectives so that it can parse the sentence "I ate spaghetti with a large silver fork."

Solution:

Add the following 3 rules

```
NOM -> ADJ NOM
ADJ -> large
ADJ -> silver
```

4. (8 points) How are polysemy and metonymy similar and/or different? Give an example of each.

Solution:

Polysemy is when a lexeme has multiple meanings. For example, "foot" (the body part) and "foot" (the unit of measurement). Metonymy is a type of polysemy where some of the multiple meanings are related concepts. For example, in "the crown announced yesterday", the word "crown" is used to refer to the person or organization (representatives) behind the crown. It is also polysemous, since the word refers to either a physical object or can mean the related concept of the monarchy or representatives of the monarchy.

5. (8 points) Suppose you are given a fully connected layer $l(x) = xW + b$ with input dimensionality d_{in} and output dimensionality d_{out}. Give the dimensionality of x, W and b in terms of d_{in} and d_{out}. (x is the input).

Solution:

The correct answer is x has dimension 1 x d_{in}, W has dimension d_{in} x d_{out} and b has dimension 1 x d_{out}.

However, we also accepted the following answers:

If you thought x was a column vector, given the convention used in the homework, we accepted interpreting the equation for the fully connected layer as $Wx + b$, in which case the answer is x has dimension d_{in} x 1, W has dimension d_{out} x d_{in} and b has dimension d_{out} x 1.

For the above, if you considered a batch size of s, then replace 1 with s.

If you thought d_{in} and d_{out} were tuples, i.e. there was d_{in_i}, d_{in_j}, d_{out_i}, and d_{out_j}, which is an unusual interpretation and not our intention, we accepted the following consistent answer: x has dimension d_{in_i} x d_{in_j}, W has dimension d_{in_j} x d_{out_j}, and b has dimension d_{out_i} x d_{out_j}, where $d_{in_i} = d_{out_i} = s$, i.e. the batch size.

6. (8 points) What is dropout? How does it work? Why does it improve learning in a neural net?

Solution:

Dropout is an effective method for preventing neural nets from overfitting. The dropout method is designed to prevent the network from learning to rely on specific weights. It works by randomly dropping (setting to 0) some percent (often half) of the neurons in the network (or in a specific layer) in each training example during stochastic gradient training. This is implemented by using masking vectors where the values are randomly set to 1 and 0 using a Bernoulli distribution and the masking vector is multiplied against the hidden layer.

3 Problem solving (38 points)

There are two problems in this section. Do both problems. Put your answers on the blank page after each question.

Recommended time: 30 minutes.

1. (19 points) **Dependency parsing.**

(a) Given the following dependency parse output, provide a diagram of the dependency tree and give the original sentence that was input (the sentence has two instances of "the").

`A = {` (5 points)

```
(<root>, PRED, are; right-arc), (are, NSUBJ, dogs; left-arc),
(dogs, DET, the; left-arc), (dogs, PREP, on; right-arc),
(on, POBJ, couch; right-arc), (couch, DET, the; left-arc),
(are, ACOMP, cute; right-arc), (cute, CC, and; right-arc),
(cute, CONJ, fuzzy; right-arc), (are, PUNCT, .; right-arc)
```

Solution:

We accepted {The dogs on the couch are cute and fuzzy., The dogs are cute and fuzzy on the couch., The cute and fuzzy dogs are on the couch.}

The first accepted answer (The dogs on the couch are cute and fuzzy.) is the preferred answer.

(b) (9 points) Show the transitions that would occur in parsing the following sentence using an arc standard parser and assuming an oracle that can tell you which transition to take next. Show the state of the stack, the buffer and the arc set (you may omit the label) as the sentence is parsed.

"Japan woke to flooded rivers."

Solution:

See the table below. Arcs are presented in the order (parent, child). The root is represented as "0", for brevity.

Stack	Buffer	Arcs	Action
0	Japan woke to flooded rivers .	$A = \emptyset$	Shift
0 Japan	woke to flooded rivers .	A	Shift
0 Japan woke	to flooded rivers .	$A = A \cup \{(woke, Japan)\}$	Left-arc
0 woke	to flooded rivers .	A	Shift
0 woke to	flooded rivers .	A	Shift
0 woke to flooded	rivers .	A	Shift
0 woke to flooded rivers	.	$A = A \cup \{(rivers, flooded)\}$	Left-arc
0 woke to rivers	.	$A = A \cup \{(to, rivers)\}$	Right-arc
0 woke to	.	$A = A \cup \{(woke, to)\}$	Right-arc
0 woke	.	A	Shift
0 woke .	$\emptyset$	$A = A \cup \{(woke, .)\}$	Right-arc
0 woke	$\emptyset$	$A = A \cup \{(0, woke)\}$	Right-arc
0	$\emptyset$	A	End

(c) (5 points) When there is a choice of actions to apply, how does the parser know which action to choose?

Solution:

We use machine learning trained on a corpus of dependency parsed text, where from the dependency parse we can encode the sequences of actions. The learning algorithm will use discrete features such as the word or POS of the word at the front of the buffer, the word or POS on top of the stack, the arcs constructed so far. The model learned on this data is applied to the new configuration to predict the next action.

(Solution to problem 1 here.)

2. (19 points) **Hidden Markov Models.** You are given the following sentence and the tables of probabilities shown in Table 1 and Table 2 below:

I want a ride on the merry-go-round.

(a) Describe how you would compute the tag transition probabilities in Table 2.

Solution:

(4 points) I would use a corpus tagged with POS. I would count the number of times the POS labeling the row comes before the POS labeling the column. I would then divide by the total number of times the POS labeling the row occurs in the corpus.

(b) Given the sentence "I want a ride on the merry-go-round." show how you would compute the probability of "ride" as a verb versus the probability of "ride" as a noun using the probabilities in Tables 1 and 2 and the Viterbi algorithm. Note that there may be other POS tags not shown here, and thus the columns and rows don't always add up to 1.

The Viterbi algorithm is shown on the next page. For this question you should:

i) Show the dynamic programming trellis at each state up to the point where "ride" is disambiguated.

ii) Show the formula with the values that would be used to compute the probability of "ride" as either verb or noun. You do not need to do the arithmetic. Just show the formula that would be computed (e.g., .05*.03*0).

	I	***want***	***ride***	***the***	***on***	***a***	***merry-go-round***
VB	0	.50	.40	0	0	0	0
TO	0	0	0	0	.99	0	0
NN	0	.40	.30	0	0	0	.99
PPSS	.50	0	0	0	0	0	0
DT	0	0	0	.99	0	.90	0

Table 1: Observation likelihoods.

	VB	**TO**	**NN**	**PPSS**	**DT**
<s>	.05	.01	.10	.40	.30
VB	.01	.20	.50	.01	0
TO	.90	0	.01	0	.01
NN	.20	.10	.30	.10	.10
PPSS	.50	.02	.10	.05	.10
DT	0	0	.80	0	0

Table 2: Tag transition probabilities. The rows are labeled with the conditioning event. Thus, P(VB|<s>) = .01.

function VITERBI(*observations* of len T, *state-graph*) **returns** *best-path*

num-states ← NUM-OF-STATES(*state-graph*)
Create a path probability matrix *viterbi[num-states+2,T+2]*
viterbi[0,0] ← 1.0
for each time step t **from** 1 **to** T **do**
  for each state s **from** 1 **to** *num-states* **do**
    $viterbi[s,t] \leftarrow \max_{1 \le s' \le num\text{-}states} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{1 \le s' \le num\text{-}states} viterbi[s',t-1] * a_{s',s}$
Backtrace from highest probability state in final column of *viterbi[]* and return path

Figure 1: The Viterbi algorithm.

(Solution to problem 2 here.)

Solution:

The dynamic trellis should include a state for each POS in the table above each word. (2 points)

We use the formula in the algorithm loop to compute the value for each node in the trellis. (2 points)

(2 points for computation at each time step)

For time step 1, the input word is "I" and the values for each node would be:

VB:1*.05 (T) *0 (O)
TO:1*.01 (T) *0 (O)
NN:1*.10 (T) *0 (O)
PPSS:1*.40 (T) *.50 (O)
DT:1*.30 (T) *0 (O)

For time step 2, the input word is "want" and the values for each node would be as shown. Note that the only state that has a non-zero value has PPSS, so that is the max from the previous paths. The value is .2, but you were not required to compute any values, so we would allow copying the formula here or providing the value, whichever you found easier.

VB: 1*.40*.50 * .5 (T) * .5 (O)
TO: 1*.40*.50 * .02 (T) * 0 (O)
NN: 1*.40*.50 * .10 (T) * .4 (O)
PPSS:1*.40*.50 * .05 (T) * 0 (O)
DT: 1*.40*.50 * .10 (T) * 0 (O)

For time step 3, the input word is "a" and the values for each node would be as follows. Here there are two non-zero paths so they must be computed and the larger path selected. The observation probabilities for "a" as any POS other than DT is 0, so we only need to consider the value at the DT node.

DT:
Path 1: 1*.40*.50*.5*.5* 0 (T) * .9 (O)
Path 2: 1*.40*.50*.10*.4 * .1 (T) * .9 (O)

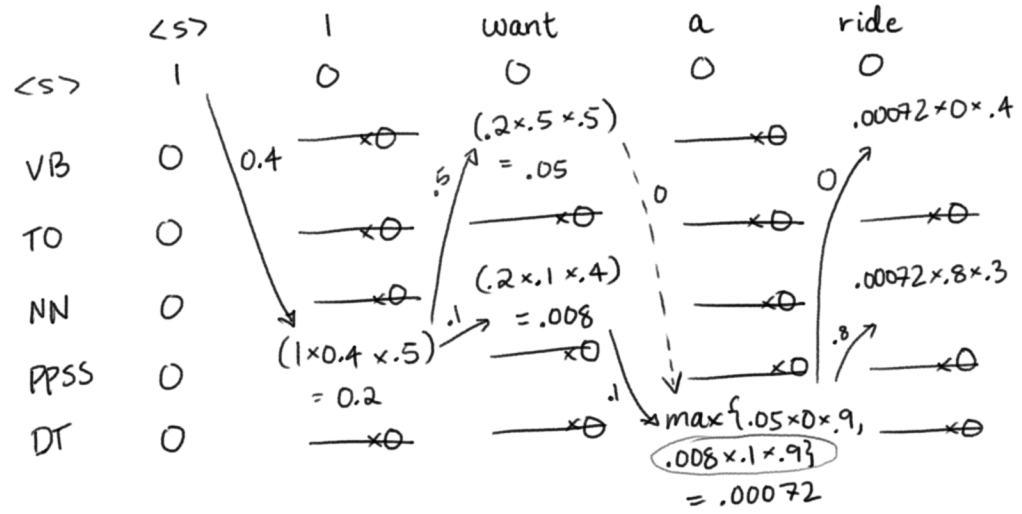


Figure 2: Nodes with an observation probability of zero are crossed out without computation.

So Path 2 is the larger.

For time step 4, the input word is "ride" and the values for each node are as shown below. Here we copy in the value for Path 2 and multiple in the transition probabilities associated with each state and observations probabilities associated with "ride" as each POS.

VB:1*.40*.50*.10*.4 * .1 * .9 * 0 (T) * .4 (O)
TO: 1*.40*.50*.10*.4 * .1 * .9 * 0 (T) * 0 (O)
NN:1*.40*.50*.10*.4 * .1 * .9 * .8 (T) * .3 (O)
PPSS:1*.40*.50*.10*.4 * .1 * .9 * 0 (T) * 0 (O)
DT:1*.40*.50*.10*.4 * .1 * .9 * 0 (T) * 0 (O)

(3 point for noting final disambiguation) Here, “ride” is disambiguated as a NN since it is the only non-zero value.