Practice questions

T/F questions

1. Tag transition probabilities in an HMM measure the likelihood of a POS tag given the preceding tag (**T)**

2. Observation likelihoods in an HMM measure the likelihood of a word given a POS tag **(T)**

3. When training a language model, the probabilities generalize well if the training corpus is too small. **(F)** (The correct answer is that it won't generalize well)

4. In gradient descent algorithm, a good strategy for setting the learning rate is to start with a large learning rate, then gradually decrease the value in each iteration **(T)**

Short answer questions

5. Describe how you would calculate the likelihood of a sentence using a bigram model:

Answer: Calculate the conditional probability of each word in the sentence given the preceding word and multiply the resulting numbers

6. What is the difference between intrinsic and extrinsic evaluation? What are the pros and cons of each?

Answer: In intrinsic evaluation, we use an evaluation metric that measures the quality of a model independent of any application. In extrinsic evaluation, we integrate the model within another application and measure how much the application improves. Extrinsic evaluation is time consuming whereas intrinsic evaluation might not reflect the models' real performance.

7. What are Morphotactics?

Answer: How and which morphemes can be affixed to a stem

8. In N-fold cross-validation, what is the percentage of data used for evaluation?

Answer: All data in the corpus are used for both training and evaluation

9. In a dependency parse tree, which nodes are labeled with words?

Answer: All nodes are labeled with words

10. Describe how a back-off model is used to alleviate the problem of 0 n-gram counts

Answer: In a back-off model, every time you had a zero count for an n-gram, you would back-off and use the counts for a lower n-gram. For example, if the count for a trigram is 0, you'd use the bigrams instead. If the bigram was 0 also, you could back off to unigram counts.

Exercises

Exercise 1:
Consider the following grammar:

S ->NP VP **VP -> Verb NP** **VP -> Verb PP** **VP ->VP PP** **NP ->NP PP** **NP -> NP and NP** **PP-> P NP**	**NP->Kathy** **NP->Connecticut** **NP->Massachusetts** **NP->November**	**Verb->drove** **P->to** **P->in** **CONJ -> and**

A. Show three parse trees that would be derived for the sentence

Kathy drove to Connecticut and Massachusetts in November

B. Trace the CKY algorithm for the sentence "Kathy drove to Connecticut"

C. [5 points] Given a treebank how would you determine probabilities for the grammar rules given in A (for use with a basic PCFG parser)?

Answer: Let's take the VP rule. There are three VP rules. I would count the total number of VP rules in the Treebank. Then, for each rule, I would count the number of times that rule occurs and divide by the total number of VP rules. That would yield the probability for each rule. I would follow a similar procedure for each rule where the same non-terminal appeared on the left-hand side.

D. Advanced probabilistic parsers make use of probability estimates other than those based on grammar rules alone, taking words into account. Describe one such probability estimate that might have helped with determining the best parse of part a.

Answer:

For the rule VP -> VP PP, I would add lexical heads, so I would get rules that looked like this:

VP -> VP (drove) PP (in),
VP -> VP (drove) PP (to),

I would do the same for the NP-> NP PP getting rules like

NP-> NP (Massachusetts) PP (in)
NP -> NP (Massachusetts) PP (in)

And so forth. Then I would calculate the probabilities associated with such rules by counting the number of times, for example, that drove occurs as a VP with the modifier PP headed by "to" following it and divide that by the total number of times that the drove occurs as a VP at all. Similarly, for the NP rules, I would count the number of times that the NP headed by Massachusetts occurs with a PP headed by in as a modifier and I would divide that by the number of times that the NP Massachusetts occurs at all.

Exercise 2:

Given the training data below, execute the following 3 steps, the first two create the HMM and the third uses this HMM for decoding: (a) calculate the likelihood probabilities for each word given each POS; (b) draw a finite state machine where states are POS and edges are labeled with transition probabilities; (c) draw a chart where the columns are positions in the sentence and the rows are names of states (start, end, POS tags) and fill in the probability scores assigned by the Viterbi algorithm assigning POS tags to the string *"flying planes"*.

Training Data:

buffalo/NNS flying/VBG is/VBZ dangerous/JJ
flying/JJ planes/NNS are/VBZ numerous/JJ
I/PRP saw/VBZ Mary/NNP flying/VBG planes/NNS
He/PRP planes/VBZ shelves/NNS

Exercise 3:

Write a regular expression that models the following languages

i. qw, qwqw, qwqwqw, …

ii. binary numbers: 0, 1, 10, 11, …,1010010, …

iii. decimal numbers: 0, 1, 2, .. 23, ..23279725, …

iv. hexadecimal numbers: 0, 1, 2, 3, 4 , 5, 6, 7, 8, 9, A, B, C, D, E, F, 12FD, FFFFFFF

Answers:

i. (qw)+
ii. (0|1[10]*)
iii. (0| [1-9][0-9]*)
iv. (0|[1-9A-F][0-9A-F]*)

Exercise 4:

Assume a bigram language model is trained on the following corpus of sentences using MLE with linear interpolation for smoothing (with the bigram λ weight set to 0.9 and the unigram λ weight set to 0.1). Since the unigram model does not need to estimate P(<s>), just completely ignore the start token when estimating the unigram model.

<s> dictator oppress people </s>
<s> army oppress people </s>
<s> dictator topple army </s>
<s> people topple army </s>
<s> army topple dictator </s>

What is the estimated probability of the following test string? Show your work. You only need to calculate the parameters of the model sufficient to solve this particular problem.

<s> people topple dictator </s>

As a hint: P(people|<s>) = 0.9(1/5) + 0.1(3/20) = 0.195