Student ID:
Name:

CS 442b-542b
Short Exam 2

Instructions: Show all the work you do. Use the back of the page, if necessary. Calculators are allowed, laptops are not allowed.

Problem 1 (20%): The training data consists of the text: "I like to play with cats, and cats like to chase my toy mouse. Cats like me, and I like cats." Ignore the sentence punctuation for this problem, that is treat the training text as one continuous sentence. Also ignore capitalization, that is "cats" and "Cats" should be treated as the same word. Suppose that we use trigram language model with maximum likelihood estimation of parameters.

(a) Compute the probability of a sentence "Cats like to play." Make sure to show all the work.

Solution:
P (Cats like to play) = P (play | cats like to)*P (to | cats like) * P (like | cats) * P (cats)
≈ P (play | like to) * P (to | cats like) * P (like | cats) * P (cats);
Let C (like to) be the count of how many times where "cats like" occurs in training data:
P (play | like to) = P (like to play) / P (like to) = C (like to play)/C (like to) = ½ = 0.5;
P (to | cats like) = P (cats like to) / P (cats like) = C (cats like to)/C (cats like) = ½ = 0.5;
P (like | cats) = P(cats like)/ P (cats) = C (cats like) / C (cats) = 2/4 = 0.5
P (cats) = C (cats) / N = 4/21 = 0.1905
P (Cats like to play) = 0.5*0.5*0.5*0.1905 = 0.0238;

(b) Compute the probability of a sentence "Black cat jumped higher than any other cat"

Solution: P (Black cat jumped higher than any other cat)
≈ P (cat | any other)*P (other | than any)*P (any | higher than)…*P (black);
P (black) = C (black)/N = 0, therefore, P (black cat jumped higher than any other cat) = 0

Problem 2 (10%): Suppose you have the following training data: "I like to play with cats, and cats like me". You decide to use add 1 smoothing (Laplace's law). What is the probability of nGram "a black cat jumps". Assume that vocabulary size is 10.
Solution: P (a black cat jumps) = [C (a black cat jumps) + 1] / (N + B) ;
C (a black cat jumps) = 0; "a black cat jumps" is a 4-gram, and there are seven 4-grams in the graining data. So N = 7;
B is the number of faked 4-grams, B = 10^4
P (a black cat jumps) = [C (a black cat jumps) + 1]/ (N+B) = 1 /10007 = 0.00009993

Problem 3 (10%): We have the following training text tagged with parts of speech: "The**\DET** time**\NOUN** flies**\VERB** very**\ADV** fast**\ADV**. Fast**\ADJ** flies**\NOUN** move**\VERB**. Time**\VERB** the**\DET** flies**\NOUN** fast**\ADV**!. Flies**\NOUN** do**\VERB** not**\ADV** fast**\NOUN**." Tag the phrase "Fast flies" using Charniak's tagger.

Solution: Let C (fast**\NOUN**) be the number of times that "fast" is tagged as a noun:
C (fast**\NOUN**) = 1; C (fast**\VERB**) = 0; C (fast**\DET**) = 0; C (fast**\ADJ**) = 1; C (fast**\ADV**) = 2, therefore the tag for "fast" is "\ADV";
C (files**\NOUN**) = 3; C (files**\VERB**) = 1; C (files**\DET**) = 0; C (files**\ADJ**) = 1; C (files**\ADV**) = 2, therefore the tag for "files" is "**\NOUN**", so the final tagging is **ADV NOUN**

Problem 4(10%): Using the same training text as in problem 3, estimate P(**VERB | NOUN**) using MLE.

Solution: Let C (NOUN) be the number of times tag "NOUN" occurs in training data, C (NOUN VERB) be the number of times that a "VERB" follows a "NOUN":
P (VERB | NOUN) = P (NOUN VERB)/P (NOUN) = C (NOUN VERB)/C (NOUN) = 3/5= 0.6

Problem 5 (25%): Suppose you are using Markov model POS tagger, and for the ith word in the sentence you are tagging, the possible part of speech labels are "ADV", "ADJ", "NOUN". For the (i+1)th word in the sentence, the possible POS tags are "NOUN" and "VERB". You have run dynamic programming for i iterations, and at the ith iteration, you have computed:
C(word i,ADJ) = 4, C(word i,ADV) = 3, C(word i,NOUN) = 9. Furthermore,

L(NOUN | ADJ) = 1	L(VERB | ADJ) = 10	L(word i+1 | VERB) = 1
L(NOUN| ADV) = 10	L(VERB| ADV) = 2	L(word i+1 |NOUN) = 3
L(NOUN|NOUN) = 3	L(VERB|NOUN) = 1	

(a) (20%) After iteration i+1, what are the values of:
C(word i+1, VERB)
C(word i+1, NOUN)
P(word i+1, VERB)
P(word i+1, NOUN)

Solution:
C(word i+1, VERB) = min{C(word i, t) + L (VERB|t)} + L(word i+1| VERB)}
C(word i, ADJ) + L(VERB|ADJ) = 4 + 10 = 14;
C(word i, ADV) + L(VERB|ADV) = 3 + 2 = 5;
C(word i, NOUN) +L(VERB|NOUN) = 9 + 1 = 10;
Therefore, C(word i+1, VERB) = C(word i, ADV) + L(VERB|ADV) + L(word i+1|VERB) = 5+1 = 6, P(word i+1, VERB) = ADV;

C(word i+1, NOUN) = min{C(word i, t) + L (NOUN|t)} + L(word i+1| NOUN)}
C(word i, ADJ) + L(NOUN|ADJ) = 4 + 1 = 5;
C(word i, ADV) + L(NOUN|ADV) = 3 + 10= 13;
C(word i, NOUN) +L(NOUN|NOUN) = 9 + 3 = 12;
Therefore, C(word i+1, NOUN) = C(word i, ADJ) + L(NOUN|ADJ)+ L(word i+1, NOUN) = 5+3 = 8, P(word i+1, NOUN) = ADJ;

(b) (5%) Suppose now that the length of the sentence is k > i+1. After (i+1)th iteration, can you tell which part of speech the (i+1)th word will get tagged with by the Markov tagger? If yes, which one, if no, why not?

Solution: We can not know any of the tags. Let's consider word j, where j < k. If we want to know the tag of word j, we should first know the tag of word j+1, say TAG J+1, then P(word j+1, TAG J+1) is the tag of word j. If we want know the tag of word j+1, we must know the tag of word j+2 …., therefore, we can not know any of the tags until we find the tag for the last word of the sentence.

Problem 6(25%) : Suppose inverted index has 4 terms and 2 documents, and the term/document counts are as follows:

$$document1 = \begin{bmatrix} term1 & term2 & term3 & term4 \\ 0 & 2 & 0 & 9 \end{bmatrix} \qquad document2 = \begin{bmatrix} term1 & term2 & term3 & term4 \\ 3 & 0 & 4 & 6 \end{bmatrix}$$

(a) (15%) Suppose that we use term counts for the weights. User query is "term3 term1 term2", that is it has one occurrence of term3, 1 occurrence of term1, and one occurrence of term2. Using the cosine similarity, if we are only asked to retrieve one document, should it be document 1 or 2? To simplify calculations, **do not normalize** vectors to length 1 for this question. Show all the work you do.

Solution: Let d1 be the vector for document 1, d2 be the vector for document 2, q be the query vector. Now we have:

d1 = [0 2 0 9], d2 = [3 0 4 6], q = [1 1 1 0]

cos (d1, q) = (d1·q)/(||d1||×||q||) = (0*1+2*1+ 0*1 +9*0)/(sqrt(4+81)*sqrt(1+1+1))

= (2)/(15.97) = 0.1252

cos(d2, q) = (d2·q)/(||d2||×||q||) = (3+4)/(sqrt(9+16+36)*sqrt(1+1+1))

= 0.52

So document 2 is closer to the query vector. Return document 2;

(b) (10%) Which term will have the highest weight in document 2 if instead of frequencies we use tf*idf for the weights? What is the value of this highest weight?

Solution:

Let w(1,2), w(2, 2), w(3, 2), w(4,2) be the weight of term 1, 2, 3 and 4 in document 2. Let tf(1,2), tf(2,2), tf(3,2) and tf(4,2) be the frequency of term 1, 2, 3 and 4 in document 2. M is the number of document, that is 2. Let df(1), df(2), df(3) and df(4) be the numbers of documents which contain term 1, 2, 3 and 4.

w(1,2) = tf(1,2) * idf(M/df(1)) = 3*log(2/1)=3*log2

w(2,2) = tf(2,2) * idf(M/df(2)) = 0*log(2/1) = 0 ;

w(3,2) = tf(3,2) * idf(M/df(3)) = 4*log(2/1) = 4*log2

w(4,2) = tf(4,2) * idf(M/df(4)) = 6*log(2/2) = 0;

The largest weight is w(3,2) = 4 * log2