National University of Computer and Emerging Sciences, Lahore Campus

Course:	Natural Language Processing	Course Code:	CS 535
Program:	MS(Computer Science)	Semester:	Spring 2019
Duration:	120 Minutes	Total Marks:	30
Paper Date:	22-March-19	Weight	20%
Section:	CS	Page(s):	6
Exam:	Midterm		

Q1) Consider a trigram HMM, as introduced in class. We saw that the Viterbi algorithm could be used to find $max_{y\,1,\,\ldots\ldots y\,n+1}\, p(x\,1, \ldots x\,n; y\,1, \ldots\ldots y\,n+1)$ where the max is taken over all sequences $y\,1, \ldots\ldots y\,n+1$ such that $y_i \in K$ for $i = 1 \ldots n$, and y_{n+1} = STOP. (Recall that K is the set of possible tags in the HMM.) In a trigram tagger we assume that p takes the form

$$p(x\,1, \ldots x\,n; y\,1, \ldots\ldots y\,n+1) = \prod_{i=1}^{n+1} q¿¿$$

Recall that we have assumed in this definition that $y_0 = y_{-1} = y_{-2} = *$, and y_{n+1} = STOP. The Viterbi algorithm is shown in figure below.

Input: A sentence $x\,1, \ldots x\,n$, paramters $q\,(s|u, v)$ and $e(x \mid s)$
Dedinitions: Define K to be the set of possible tags. Define $K_{-1} = K_0 = \{*\}$, and $K_k = K$ for $k = 1 \ldots n$.
Initialization: *Set* $\pi\,(0,*,*) = 1$
Algorithm: For $k = 1 \ldots n$,

For $u \in K_{k-1}, v \in K_k$,

$$\pi\,(k,u,v) = max_{w \in K_{k-2}} \left(\pi\,(k-1,w,u) \times q\,(v \mid w, u) \times e\,(x_k \mid v) \right)$$

Return $max_{u \in K_{n-1}, v \in K_n} \left(\pi\,(n,u,v) \times q\,(STOP \mid u, v) \right)$

Now consider a **five-gram** tagger, where p takes the form

$$p(x\,1, \ldots x\,n; y\,1, \ldots\ldots y\,n+1) = \prod_{i=1}^{n+1} q¿¿$$

We have assumed in this definition that $y_0 = y_{-1} = y_{-2} = y_{-3} = *$, and y_{n+1} = STOP. In the box on next page, give a version of the Viterbi algorithm that takes as input a sentence $x\,1, \ldots x\,n$, and finds $max_{y\,1,\,\ldots\ldots y\,n+1}\, p(x\,1, \ldots x\,n; y\,1, \ldots\ldots y\,n+1)$ for a **five-gram** tagger, as defined above equation

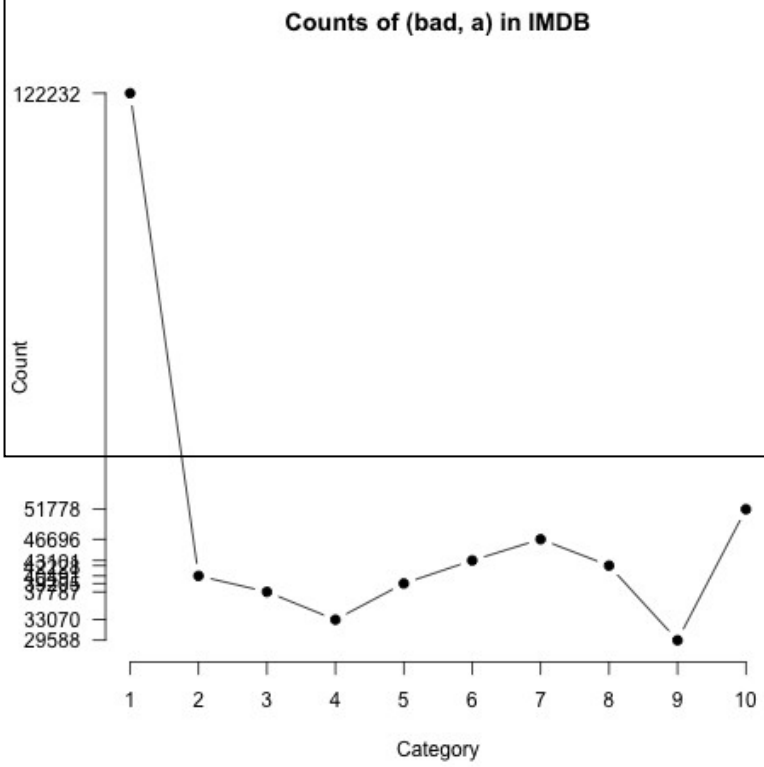
Input: A sentence $x\,1, \ldots x\,n$, paramters $q\,(w|r, t, u, v)$ and $e(x \mid s)$
Dedinitions: Define K to be the set of possible tags. Define $K_{-3} = K_{-2} = K_{-1} = K_0 = \{*\}$, and $K_k = K$ for $k = 1 \ldots n$.

Q2) Why “stupid backoff” smoothing method is called “stupid backoff” ? What is stupid about this method. [2 Marks]

Solution:

This method does not produce probability distributions after smoothig and produces scores.

Q3) Following plot shows polarity of word ”bad” in IMDB reviews. X-axis shows rating of reviews (1 means poor and 10 means best). Y-axis shows count of the word “bad” in each category of review. Count of word “bad” is higher in reviews with rating 10 as compared to reviews with rating less rating (e.g. 9, 8, 7, 4). What is reason for this unexpected behavior? (word “bad” should have lower count in reviews with best rating 10). Also state solution to resolve this issue. [2 Marks]



Solution:

The reason can be higher number of reviews for class with review rating 10. The solution is to normalize the count by dividing the count of word with total number of words in that class.

Q4) Represent words “apple” and “cycle” as vectors using following corpus. Use TF.IDF weights. Assume the words occurring 2 positions before and after the word make its context. E.g. context words for the word “ride” are “like, to, cycle, often”. Number of dimensions of each vector should be according to following corpus. [6 Marks]

I like to ride cycle often.
We ate apple and oranges.
We ate apple not oranges.

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$\mathrm{idf}_i = \log\left(\frac{N}{\mathrm{df}_i}\right)$$

Solution:

Tf.IDF of ride = 1*log (3/1) = 0.48

Tf.IDF of ate = (1 + log 2)*log (3/2) = 1.3* 0.176 = 0.23

	I	like	to	ride	often	We	ate	and	oranges	not
Apple	0	0	0	0	0	0.23	0.23	0.48	0.23	0.48
cycle	0	0	0.48	0.48	0.48	0	0	0	0	0

Q5) Multiple Choice questions: [3 Marks]

1. In an HMM, tag transition probabilities measure
    a. The likelihood of a POS tag given a word
    b. The likelihood of a POS tag given the preceding tag
    c. The likelihood of a word given a POS tag

Solution: b

2. In an HMM, observation likelihoods measure
    a. The likelihood of a POS tag given a word
    b. The likelihood of a POS tag given the preceding tag
    c. The likelihood of a word given a POS tag
    d. The likelihood of a POS tag given two preceding tags

    Solution: c

3. To compute the likelihood of a sentence using a bigram model, you would:
    a. Calculate the conditional probability of each word given all preceding words in a sentence and multiply the resulting numbers
    b. Calculate the conditional probability of each word given all preceding words in a sentence and add the resulting numbers
    c. Calculate the conditional probability of each word in the sentence given the preceding word and multiply the resulting numbers
    Calculate the conditional probability of each word in the sentence given the preceding word and add the resulting numbers.

Solution: c

Q6) What is advantage of using continuation count in Kneser Ney smoothing instead of unigram probability of the word. Illustrae with example. [2 Marks]

Solution:

Context of word is considered in continuation count. For example if we want to kow which word has high probability to occur after the word ”reading” then we can consider continuation count of different words. Continuation count is count of how many times a word appears after a new word. The word ”glasses” has

higher continuation count than the word ”Francisco” but ”Fransisco” is more common than glasses. Reading glasses makes more sense than reading Fransisco.

Q7) You are an English teacher and you ask your class to write a play in the style of Shakespeare. You want to score their plays using a trigram language model you computed from a corpus of all Shakespeare plays but you find that the data is too sparse and most of your students’ sentences receive a score of zero. How would you use a back-off model to alleviate this problem? [3 Marks]

Solution:

I can use bigram model. If a bigram has zero probability then I can use unigram model.

Q8) Consider the following grammar: [4 Marks]

S ->NP VP VP -> Verb NP VP -> Verb PP VP ->VP PP NP ->NP PP NP -> NP CONJ NP PP-> P NP	NP->Ali NP->Lahore NP->Karachi NP->October	Verb->drove P->to P->in CONJ -> and

Show the parse trees that would be derived for the sentence *Ali drove to Lahore and Karachi in October*

Q9) Given a treebank how would you determine probabilities for the grammar rules given in Question 8 (for use with a basic PCFG parser)? [2 Marks]

Solution:

Using count of grammar rules being used in Treebank as follows:

Prob (A → B) = Count (A → B) / Count (A)