Exam 1

CS 322: Natural Language Processing

Fall 2015

Name: ______________________________

Read all of the following information before starting the exam:

- Show all work, clearly and in order, if you want to get full credit. I reserve the right to take off points if I cannot see how you arrived at your answer (even if your final answer is correct).
- Justify your answers algebraically whenever possible to ensure full credit. When you do use your calculator, sketch all relevant graphs and explain all relevant mathematics.
- Circle or otherwise indicate your final answers.
- Please keep your written answers brief; be clear and to the point. I reserve the right to take points off for rambling and for incorrect or irrelevant statements.
- This exam has 5 problems and is worth 71 points.
- Good luck!

1	2	3	4	5	Σ
8	21	20	10	12	71

1. Short Answer (*8 points*) For each of the following questions, give *brief* (1-2 sentences) answers.

a. (*4 pts*) Define *stemming.*

b. (*4 pts*) What is the Markov assumption? Specifically, how does it apply to building language models?

2. Language Models (*21 points*) Consider the following sample training corpus:

```
I will not eat them in a box.
I will not eat them with a fox.
```

For the following sentence,

```
I will _______
```

a. (*6 pts*) Unigram LM
What is the unigram probability for the next word being not?

What is the unigram probability for the next word being eat?

b. (*6 pts*) Unsmoothed Bigram LM
(For this model, use beginning- and end-of-sentence tokens in your calculations.)
What is the (unsmoothed) bigram probability for the next word being not?

What is the (unsmoothed) bigram probability for the next word being eat?

c. (*6 pts*) Smoothed Bigram LM
What is the smoothed (LaPlace smoothing) bigram probability for the next word being not?

What is the smoothed (LaPlace smoothing) bigram probability for the next word being eat?

d. (*3 pts*) Suppose that a training set has 4 words in its vocabulary. Three of the words appear 20% of the time each, the last word appears 40% of the time. What is the perplexity of a unigram model trained on this corpus?

3. Edit Distance (*20 points*) Due to an incident involving bees, honey theft, and an active immune system, Professor W. P. has had the fingers on his left hand swell up to a very large size. The result of this unfortunate situation is that when he types on the keyboard, he is extremely likely to "mash" keys with his left hand.
As an example, I got an e-mail from him with this line:
"Hrello, howe asrwe you?"
A simple analysis of many e-mails from him shows that every time he types a letter with his left hand (letter B, G, T and leftward) he has a 50% chance of also mashing another letter.
Consider this code for the Edit Distance algorithm that is used to help assist in spelling correction on Professor W. P.'s computer:

```python
def dist(m, c):
    len_m = len(m)
    len_c = len(c)

    # set up matrix
    dist_matr = [([0] * (len_c+1)) for i in range(len_m+1)]

    dist_matr[0][0] = 0
    for mi in range(len(dist_matr)):
        dist_matr[mi][0] = mi
    for ci in range(len(dist_matr[0])):
        dist_matr[0][ci] = ci

    for mi in range(1, len(dist_matr)):
        for ci in range(1, len(dist_matr[0])):
            sub_c = dist_matr[mi-1][ci-1] + cost_sub(m[mi-1], c[ci-1])
            ins_c = dist_matr[mi][ci-1] + cost_ins(c[ci-1])
            del_c = dist_matr[mi-1][ci] + cost_del(m[mi-1])
            dist_matr[mi][ci] = min(sub_c, ins_c, del_c)

    return dist_matr[len_m][len_c]

def cost_ins(ch):
    return 1

def cost_del(ch):
    return 1

def cost_sub(ch_m, ch_c):
    if ch_m == ch_c:
        return 0
    else:
        return 2
```

a. (*10 pts*) How would you alter this code so that it is better at correcting for Professor W. P.'s situation? Briefly (2-3 sentences) describe your plan, then include any necessary code changes.

b. (*10 pts*) After solving Professor W. P.'s problem, you are asked by another Professor K. R. to assist with a slightly different situation. Professor K. R. has a mild form of dyslexia that manifests itself such that she will often swap two letters in a word and not notice. Otherwise, she is an impeccable typist. An analysis of some of her e-mails shows that her misspelled words have an extra letter 5% of the time, a missing letter 10% of the time, and swapped letters 85% of the time.
How would you alter the original code so that it is better at correcting Professor K. R.'s mistakes? Briefly (2-3 sentences) describe your plan, then include any necessary code changes.

4. POS Tagging (*10 points*) Consider the following training set for a POS tagging system.

```
it/PRP 's/VBZ close/JJ to/TO midnight/JJ and/CC something/NN
evil/NN 's/POS lurking/NN in/IN the/DT dark/JJ

under/IN the/DT moonlight/NN you/PRP see/VBP a/DT sight/NN
that/WDT almost/RB stops/VBZ your/PRP$ heart/NN

you/PRP try/VBP to/TO scream/VB but/CC terror/NN takes/VBZ the/DT
sound/NN before/IN you/PRP make/VBP it/PRP
```

Use the assumption that $P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$.

Also use the bigram assumption for tags and words. Show how you got each answer.

a. (*5 pts*) Based on this training set, calculate the probability of the tag sequence `<s> IN DT JJ </s>`

b. (*5 pts*) Based on this training set, calculate the probability of the tag sequence `<s> IN DT JJ </s>` for the word sequence `<s> under the dark </s>`.

That is, calculate $P($`<s> IN DT JJ </s>`$|$`<s> under the dark </s>`$)$

5. Regular Expressions (*12 points*) For the following regular expressions and the descriptions of their intent, give an example of a false positive and a false negative, or say that there is none. Make the following assumptions:

1. The search text is entirely lowercase letters and spaces, with no punctuation of any sort.
2. The search text may contain multiple lines.
3. For each expression, the capturing group (in parentheses) is used to capture the stated intent, the rest of the expression is for matching context.

a. (*4 pts*)
Words that end in -ed
`\b([a-z]*ed)\b`

b. (*4 pts*)
Words with 3 or more consecutive vowels
`\b([a-z]+[aeiou]{3,3}[a-z]+)\b`

c. (*4 pts*)
Words that follow words that end in x or z.
`\b[a-z]*[xz]\s*([a-z]+)\b`