CSE 5539-0010 Social Media and Text Analysis
Quiz #1: Background Review

Alan Ritter, The Ohio State University

Due: Beginning of class, Thursday, January 18th, 2018

Your Name: ________________________________ OSU Username: _______________________

Instructions This quiz will not be graded; you get credit (2 points) for simply submitting the answers to these questions. The goal of this quiz is for you to determine whether you have the mathematical and programming background needed to take this class (you may audit the class instead). Please submit your completed quiz at next class. You do not have to know Python or Numpy package to take the class, but you should feel comfortable to self-teach yourself Python programming in two weeks and simple Numpy functions in about five weeks in order to complete the homework assignments successfully.

1 Probability Review

For events A and B, prove $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$.

2 Calculus Review

The *sigmoid* function is defined as $\sigma(x) = \frac{1}{1+e^{-x}}$. Prove that the gradients of the sigmoid function can be rewritten as the function itself, $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. (Hint: use the chain rule of calculus derivatives)

3 Python Programming Review

What does the regular expression “#[a-zA-Z0-9_]+” do?

4 Python Programming Review (Numpy package)

The *softmax* function is defined as $softmax(\mathbf{x})_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$. And conveniently, softmax is invarint to constant offsets in the input, that is, for any input vector $\mathbf{x}$ and any constant c, $softmax(\mathbf{x}) = softmax(\mathbf{x} + c)$ where $\mathbf{x} + c$ means adding the constant c to every dimension of $\mathbf{x}$. Can you prove it?

Show your efficient implementation of the softmax function in Python below (program on your computer first then write down code here). You will find numpy functions np.exp, np.sum, np.reshape, np.max, and numpy broadcasting useful. Numpy is a Python package for scientific programming. What is your output?

```
import numpy as np

def softmax(x):

    # Handle the special case: when vector x is only 1-dimensional
    if x.ndim <= 1:
        x = x - np.max(x)
        ex = np.exp(x)
        return ex / np.sum(ex)

    ### YOUR CODE HERE  -- for vectors that are not 1-dimensional










    ### END YOUR CODE

    return dist

# Check your Softmax implementation
print softmax(np.array([[101,102],[-1,-2]]))
```

(Hint: The outputs for softmax(np.array([[1,2],[3,4]])) is [[0.2689, 0.7311], [0.2689, 0.7311]].)