**CS 4740 (+crosslists): Natural Language Processing** **(Due: January 31, 11.59 p.m.)**

# Homework #0

Instructors: Claire Cardie, Tanya Goyal

**Course Policy**: Read all the instructions carefully before you start working on the assignment, and before you make a submission.

- You should do this assignment **individually**. Do not consult your peers or generative AI engines for any part of this assignment.
- This assignment is designed to test whether you have the necessary background for this course. It should take you around 2.5 hours to complete this assignment. If you find yourself struggling with a large portion of this assignment, please talk to the course staff about whether this course is appropriate for you.
- This assignment is worth 3% of your grade. Make sure to submit this assignment by the due date mentioned above. **You may not use any slip days for this assignment.**
- You will submit the assignment on gradescope. You do not have to submit any code for this assignment. Only submit a pdf with answers to both Section 1 and Section 2.
- The course staff will not check the correctness of your submission for this assignment. All students who turn in a *reasonable* submission will get the full grade on this. However, make sure you compare against the answer key when it is released to self-evaluate your readiness for the course.
- The University Academic Code of Conduct will be strictly enforced.

## Section 1: Math Recap

(10 points)

**(1)** (1 point) Given two matrices $\mathbf{A} \in \mathbb{R}^{X \times Y}$ and $\mathbf{B} \in \mathbb{R}^{Z \times Y}$ where $X \neq Y \neq Z$. Which of the following operations are invalid? If the operations are valid, what is the dimensions of the final matrix?

***(Solution)***

(a) $A^T B$
Invalid

(b) $A^T A B^T B$
$Y \times Y$

(c) $A A^T B B^T$
Invalid

(d) $A^T A B^T$
$Y \times Z$

**(2)** (1.5 + 1.5 = 3 points) The joint probability distribution of two random variables X and Y is:

| P(X,Y) | X=1 | X=2 | X=3 |
|---|---|---|---|
| Y=1 | 0.2 | 0.1 | 0.1 |
| Y=2 | 0.1 | 0.3 | 0.2 |

(a) Compute the value of $E[X]$. Hint: First compute marginal probability $P(X)$.

***(Solution)***
$P(X=1) = \sum_Y P(X,Y) = 0.2 + 0.1 = 0.3$

P(X=2) = 0.1 + 0.3 = 0.4

P(X=3) = 0.1 + 0.2 = 0.3

$E[X] = 1 * P(X=1) + 2 * P(X=2) + 3 * P(X=3) = 0.3 + 0.8 + 0.9 = 2$

(b) Compute the value of $E[X+Y]$.

***(Solution)***
$P(Y=1) = 0.2 + 0.1 + 0.1 = 0.4, P(Y=2) = 0.1 + 0.3 + 0.2 = 0.6$

E[Y] = 1*P(Y=1) + 2*P(Y=2) = 0.4 + 1.2 = 1.6

$E[X+Y] = E[X] + E[Y] = 2 + 1.6 = 3.6$ (using linearity of expectation)

**(3)** (1.5 + 1.5 = 3 points) You have two bags. Bag A has 2 red balls and 3 blue balls, bag B has 3 red balls and 7 blue balls. You randomly draw one ball from a random bag.

(a) What is the probability that the drawn ball is red?

***(Solution)***
$P(Red) = P(Red|A) * P(A) + P(Red|B) * P(B)$

$P(Red) = \frac{2}{5} \times \frac{1}{2} + \frac{3}{10} \times \frac{1}{2} = \frac{7}{20}$

(b) Suppose the drawn ball is red. What is the probability that you drew the ball from Bag A?

***(Solution)***
We need to find $P(A|Red)$

$P(A|Red) = \frac{P(A,Red)}{P(Red)} = \frac{P(Red|A) \times P(A)}{P(Red)}$

$P(A|Red) = \frac{\frac{2}{5} \times \frac{1}{2}}{\frac{7}{20}} = \frac{4}{7}$

**(4)** (1 + 2 = 3 points) We define the following function:

$$f(x) = \frac{1}{1+e^{-x}}$$

(a) Prove $f(x) + f(-x) = 1$

***(Solution)***
$f(x) + f(-x) = \frac{1}{1+e^{-x}} + \frac{1}{1+e^{x}} = \frac{1}{1+\frac{1}{e^{x}}} + \frac{1}{1+e^{x}} = \frac{e^{x}}{1+e^{x}} + \frac{1}{1+e^{x}} = 1$

(b) Prove $\frac{\partial f}{\partial x} = f(x)(1 - f(x))$

***(Solution)***
$\frac{\partial f}{\partial x} = \frac{-(-e^{-x})}{(1+e^{-x})^2} = \frac{e^{-x}}{(1+e^{-x})^2}$
$1 - f(x) = 1 - \frac{1}{1+e^{-x}} = \frac{e^{-x}}{1+e^{-x}}$

$f(x)(1 - f(x)) = \frac{1}{1+e^{-x}} \times \frac{e^{-x}}{1+e^{-x}} = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{\partial f}{\partial x}$

**Section 2: Coding** (20 points)

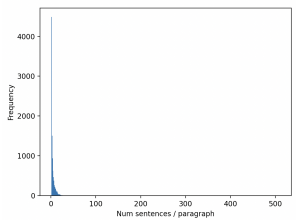
In this section, you will implement two kinds of tokenizers: (1) a **sentence tokenizer** that splits a paragraph or document of text into individual sentences, and (2) a **word tokenizer** that splits text into individual words.
With this assignment, you are given a text file `rawtext.txt`. **You can download this file at** `https://www.cs.cornell.edu/courses/cs4740/2025sp/hw0/rawtext.txt`. Each line in this file contains one paragraph from Wikipedia about a famous person. The document is lowercased. We will use this file for our investigation of different tokenization schemes.

**You will not submit code.** Just run the code you write and answer the following questions. Include any graphs and analysis you conduct using your code.

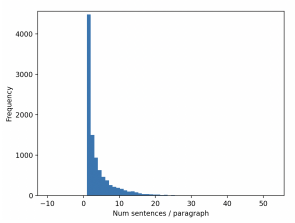
**(1)** First, we will implement a simple sentence tokenizer. For English, we can split a paragraph on certain punctuations like `.?!` etc. (think whether there are other punctuations that need to be included in this set) that occur at the end of a sentence. Write a python function that splits a paragraph into sentences.

(a) (1+ 2 = 3 points) Use your sentence tokenizer to tokenize each paragraph of the `rawtext.txt` file into sentences. Report the average number of sentences per paragraph? Plot a histogram of the number of sentences in a paragraph for the corpus.

***(Solution)***
The plot might look slightly different based on your implementation. Our implementation resulted in the following plot (Fig 1a). We also show another plot that zooms in on the graph corresponding to less than 50 sentences/paragraph (Fig 1b).




(a) Number of sentences / paragraph (full histogram)




(b) Number of sentences / paragraph (zoomed in histogram)

Average number of sentences / paragraph = 4.46

(b) (2 points) Can you think of scenarios where naively splitting a sentence on punctuations is not an ideal strategy? What other constraints will help? Discuss.

***(Solution)*** There are several possible answers here. One scenario is a sentence which contains a quote: *John said "I went to the market. The shops were closed".* In this case, splitting on "." will split the quote. Ideally, our sentence tokenizer should treat the above as one sentence. There might be "." used in abbreviations (e.g. Dr.), links (www.example.com), etc.

(c) (2 points) Re-do part (a) using `nltk`'s sentence tokenizer instead of your own implementation. Do you observe any differences? Give 2 examples where sentence tokenization for a given paragraph is different and discuss why.
Instructions to install `nltk` are at: `https://www.nltk.org/install.html`.

```
import nltk
nltk.download('punkt_tab') # you only need to do this once
paragraph = "x and y went to the market . they bought eggs and bread ."
sentences = nltk.sent_tokenize(paragraph)
```

***(Solution)***
Average number of sentences / paragraph = 3.85

*theodorus bailey (october 12, 1758september 6, 1828) was an american lawyer and politician from poughkeepsie, new york, who represented new york in both the u.s. house and senate.*

The above is one sentence according to nltk, but the naive sentence tokenizer splits it on "u.s. ". You can easily find other similar examples.

(c) (1 point) Did you notice any cases where nltk's tokenizer was incorrect?

***(Solution)*** You can essentially give any example of an error you see here. We show one example below. Note than we we lower cased all text in the dataset we provide. This means that nltk can no longer use important cues like the fact that a "." followed by a capital letter is likely a sentence boundary, etc.

Example: *she won the british open in 1978, beating her fellow australian player vicki hoffman in the final 9-4, 9-7, 9-2. newman was also runner-up at the british open in 1976, when she lost in the final to australia's heather mckay.* NLTK does not split on the "9-2." and treats this as one sentence.

**(2)** Next, we will use `nltk`'s word tokenizer to compute some statistics for our text corpus.

(a) (1 point) Iterate through the sentences you tokenized in the previous step (using `nltk`'s sentence tokenizer) and report the mean number of words/sentence.

(b) (2 point) In a number of NLP tasks, one usually removes `stopwords` from a text corpus before using it for downstream tasks. You can view `nltk`'s English language stopwords by:

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

Can you think of reasons why stopwords are removed before performing certain kinds of text analysis? Think of the sentiment classification problem from lecture 1. Are there tasks where stopwords should not be removed?

***(Solution)*** We will accept several answers here. Suppose my task is to build a sentiment classifier and my strategy is to use the unigram words in the text as my features. *It was a great movie.* Here, stop words like *a, was*, etc. do not give meaningful signals for the task at hand and can be removed. On the other hand, for the sequence tagging tasks we discussed in class, it is important to retain the stopwords.

(c) (2 points) List the 20 most frequent words in the whole corpus (not considering stopwords or punctuations) and their counts. Are there any surprise entries in this list?

***(Solution)***

{born: 6765, 's: 4790, played: 2901, also: 2790, american: 2520, football: 2285, first: 2183, league: 1903, university: 1785, new: 1727, known: 1713, former: 1663, one: 1659, team: 1611, player: 1575, world: 1551, professional: 1541, national: 1523, member: 1508, career: 1289}

**(3)** We know that each line in the file `rawtext.txt` contains part of a Wikipedia biography.

(a) (1 point) Looking at the top 200 most frequent words in the dataset, can you guess which professions are most frequent in the dataset? Give examples of 3 professions.

***(Solution)*** We will accept multiple answers here. Football players, baseball players, directors, actors, etc.

(b) (3 points) For each profession you identify, also list 3 other words that co-occur most frequently with the profession word. Two words co-occurring means that they are both mentioned in the same paragraph at least once. E.g. if "actor" is a profession you identify, maybe the most frequently co-occurring words would be "films" or "television".

***(Solution)***

Depends on the three professions you choose, but essentially, you need to re-run your code for part 2.c. for each profession, but only considering the subset of paragraphs that mention that profession word.

(b) (3 points) Let $P(x)$ be the probability that word $x$ occurs at least once in a paragraph. For the profession and related word pairs you identified in part (b), compute $P(< \text{profession} > | < \text{related word} >)$.

***(Solution)***

Depends on the profession and related words you chose in part (b). Your code should, for each (profession, related word pair), iterate through the paragraphs and count (1) number of paragraphs that contain both the profession word and the related word, (2) number of paragraphs that contain the related word

Finally, compute $\frac{\text{count(paragraphs that include both the profession word and the related word)}}{\text{count(paragraphs that contain the related word)}}$.