CSCI 59000 NLP Spring 2024 Homework 1

Edwin Sanchez

January 25th, 2024

Problem 1 (10 points)

Provide answers to the following operations. If it is not possible to calculate, write "invalid."

(a)

$$\begin{bmatrix} 2 & 7 & 7 \\ 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

Dimensions: 2x3 X 3x2 → 2x2.

$$\begin{bmatrix} 14 & 11 \\ 4 & 9 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 0 & 6 \\ 9 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 7 \\ 2 & 3 \\ 5 & 1 \end{bmatrix}^{\top}$$

Dimensions: 2x3 X 3x2 → 3x3.

$$\begin{bmatrix} 42 & 18 & 6 \\ 23 & 24 & 47 \\ 9 & 7 & 6 \end{bmatrix}$$

(c)

$$\begin{bmatrix} 2 & 0 & 2 \end{bmatrix}^{\top} \begin{bmatrix} 4 & 1 & 4 \\ 4 & 4 & 0 \\ 1 & 0 & 3 \end{bmatrix}^{\top}$$

Dimensions: 3x1 X 3x3 → DNE.

(d)

$$\begin{bmatrix}3 & 1 & 0\end{bmatrix}^{\top}\begin{bmatrix}1 & 4 & 5\end{bmatrix}$$

Dimensions: 3x1 X 1x3 → 3x3.

$$\begin{bmatrix}3 & 12 & 15\\ 1 & 4 & 5\\ 0 & 0 & 0\end{bmatrix}$$

(e)

$$\begin{bmatrix}1 & 4\\ 7 & 0\end{bmatrix}^{\top}\begin{bmatrix}6 & 8\end{bmatrix}$$

Dimensions: 2x2 X 1x2 → DNE.

Problem 2 (20 points)

Write regular expressions for the following cases.

(a) One or more words (only with lowercase alphabets) separated by spaces
e.g., "red blue green white"

- /**[a-z]**/: Any (only) lowercase letters.
- /**([a-z])+**/: Any (only) lowercase letters, but at least one or more letters.
- $/\backslash b([a-z])+\backslash b/$: Any (only) lowercase letters, but at least one or more, with word boundaries around the letter or letters.
- $/(\backslash b([a-z])+\backslash b)+/$: Any (only) lowercase letters, but at least one or more, with word boundaries around the letter or letters. All repeated one or more times.

Full Finished Statement: $/(\backslash b([a-z])+\backslash b)+/$

(b) Title case sentences, assuming all words are capitalized. Note that the text can contain numbers and punctuation.
e.g., "Why Sleep Is So Important To Your Health?"

- $/\backslash b([a-z])+\backslash b/$: Words surrounded by word boundaries.
- $/\backslash b[A-Z]([a-z])*\backslash b/$: Words surrounded by word boundaries. No longer needs + as we're forcing at least one character with the statement $[A-Z]$.
- $/\backslash b[A-Z]([a-z0-9]|\backslash W)*\backslash b/$: Words with capital letters to start that are surrounded by word boundaries. Additionally, there can be non-alphanumeric characters at any point in the word or words.

Full Finished Statement: $/\backslash b[A-Z]([a-z0-9]|\backslash W)*\backslash b/$

(c) Strings that contain the word "ice" without matching the words that contain "ice" such as "icecream" or "ice-bucket"

- $/\backslash b(ice)\backslash b/$: Finds the word "ice" with word boundaries.
- $/(\backslash b|\wedge)(ice)(\backslash b/|\$)/$ Finds the word "ice" with word boundaries, or when ice starts or ends the sentence.

Full Finished Statement: $/(\backslash b|\wedge)(ice)(\backslash b/|\$)/$

(d) The set of all lowercase alphabetic strings ending in a "b"

- $/([a-z])*b/$: This gets all of the lowercase letters or no letters prepending a word, and then ending in the letter b..

Full Finished Statement: $/([a-z])*b/$

(e) All strings that start at the beginning of the line with an integer and that end at the end of the line with a word

- $/\wedge[0-9]/$: Any string that starts with any number.
- $/\wedge[0-9](.*)/$: Any string that starts with any number, followed by any number of characters (any character), or none at all.
- $/\wedge[0-9](.*)\backslash b([a-zA-Z]*)\$/$: Any string that starts with any number, followed by any number of any character. Must end with a word bounded by a word boundary on the left, and the end of the line on the right, with any number of lowercase or uppercase letters.

Full Finished Statement: $/\wedge[0-9](.*)\backslash b([a-zA-Z]*)\$/$

Problem 3 (20 points)

(a) Pick a language other than English, which you are familiar with. Your mother tongue will be good if it is not English. If your mother tongue is English only, you can pick any language you are most familiar with.

i. Which language did you pick?
I picked **Spanish**.

ii. What kind of challenges do you expect when processing texts in that language and building NLP models? List two or three challenges. Describe each challenge concisely. If you use an example, please translate it into English.

- Spanish verbs have many cases where the verb's root is modified based on who it is directed towards. One example is Tener, spanish for "to have". Tener is conjugated as Tengo for "I have", but changes to Tienes for "You have". For the phrase "We have", it becomes Tenemos. Finally for "They have", it becomes Tienen. With this we can see how the root flips back and forth between Ten and Tien. This is far from the only word that does this in spanish, making it hard to isolate the root of a word.
- Conjugation also makes tasks like Stemming difficult as the suffixes are not always suffixes on all words. Consider any word that ends with "-go". Many are verb conjugations, but some are also nouns, such as "Mango".
- Spanish also contains some extra special characters, such as the upside down question mark to begin questions in text: ¿. While not unparsible, one would need to remember to account for these extra characters, in comparison with English. This also includes letters with accents, such as the accented letter "o": ó. You would need to make sure to include these characters for words as they are used frequently.

(b) Find one publicly available English dataset for an NLP task. Describe the following attributes.

The dataset I have found is called the Twitter US Airline Sentiment Dataset.

Link: `https://www.kaggle.com/datasets/crowdflower/twitter-airline-sentiment`.

i. Motivation (e.g., why was the corpus collected? by whom? who funded it?)

This corpus was collected to do sentiment analysis, specifically on tweets (from twitter) about US airlines. The original dataset creators

are CrowdFlower (now known as Figure Eight Inc.), a crowd-sourcing dataset company. They are now owned by Appen, an AI company.

ii. Situation: In what situation was the text written?

The text was written by active twitter users in 2015. All of the tweets are selected for their focus on US Airlines.

iii. Collection process: if it is a subsample how was it sampled? Was there consent? What kind of pre-processing was done?

The samples were pulled from publicly available tweets. There is no guarantee that these are all of the tweets from that time period. There wasn't consent as these were pulled from a publicly accessible website, where the tweets were made public. The dataset is organized into an sql database.

iv. Annotation process: is it annotated? If so, which labels? what was the annotation process?

Yes, the dataset is annotated. The labels are as follows:

- tweet_id
- airline_sentiment
- airline_sentiment_confidence
- negativereason
- negativereason_confidence
- airline
- airline_sentiment_gold
- name
- negativereason_gold
- retweet_count
- text
- tweet_coord
- tweet_created
- tweet_location
- user_timezone

The annotation process consisted of first determining if a given tweet was positive, negative, or neutral, and then identifying the negative reasons.

Problem 4 (Programming involved, 50 points)

Download the Diplomacy training dataset (train.jsonl): `https://sites.google.com/view/qanta/projects/diplomacy`

We will only use the language data in the "messages" field in the JSON format. First, read the dataset (train.jsonl) and extract only the "messages" using a JSON parser. Write a new file ("data.txt") which contains messages in each line. We will use **data.txt** from now on.

Use the NLTK package (`https://www.nltk.org/`) to split the data into sentences. Use the `sent_tokenize()` function.

(a) How many sentences are there? Note that you need to ignore empty lines and empty sentences.

There are 17,901 sentences (ignoring the empty lines and sentences.)

Now let's find words. First, split the sentences using the python `split(' ')` function.

(b) How many tokens are there?

Splitting by spaces, there are 266,717 tokens.

This time, use NLTK's `word_tokenize()` function to split into words.

(c) How many tokens are there now?

Using the `word_tokenize()` function, there are 321,201 tokens.

Lowercase all the words.

(d) How many tokens and types now?

Now there are 321,096 tokens, and 8,813 types (our vocabulary).

(e) Compare the number of tokens from (b), (c), and (d). Why are they different?

The first count splits based on spaces. `word_tokenize()` works in much the same way, except it separates out punctuation as its own tokens. This is why we see the drastic increase in the token count. Splitting by spaces ignores punctuation that is not spaced away from the words, while the tokenize function counts each and every punctionation as its own token. Since punctuation is used heavily, this makes the tokenize function's output a much larger list. For reference, there's 12,822 instances of the /./ (period) punctuation.

But when we look at the lowercase text, our word count drops by about a hundred. This is due to words that simply had different cased letters (but being the same word) being counted as different tokens. Making the text lowercase negates this.

Lastly, make a dictionary of word type counts. The dictionary contains word type as its key, and frequency as its value. Sort the dictionary.

(f) What is the most frequent word type?

The most frequent word type is: 'i', with 13,052 instances.

(g) What is the 5th most frequent word type?

The most frequent word type is: 'you', with 9,412 instances.

(h) Using the sorted dictionary, draw a graph to see if this data shows the Zipf's law. X-axis: ranked words, Y-axis: frequencies. Submit your graph and discuss it (in your pdf). Submit your code as well.

The graphs appear on the next page.

Based on the graphs, it is my conclusion that Zipf's Law does make an appearance in this dataset. After normalizing the frequencies and then drawing Zipf's Law's function in proportion to our frequency data, we can see that the frequencies drop steeply at first and then slowly levels out for the rest of the words. I think that more data would increase the potency of this figure, as the differences would manifest more intensely.

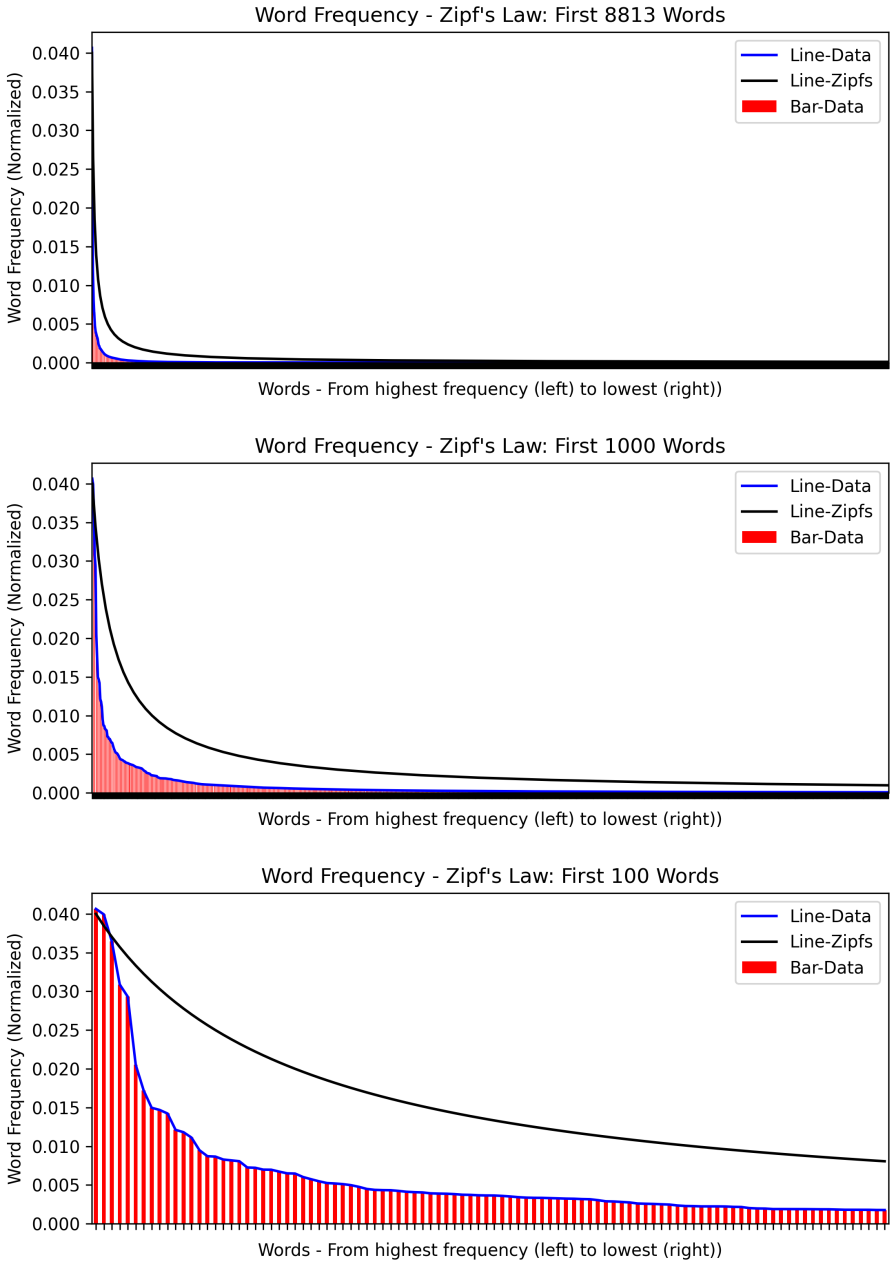

Figure 1: The normalized word frequencies of the Diplomacy Training Dataset, in bar form and in line form (blue line). Zipf's Law's Proportion (**black** line) is also displayed for comparison.