IST 664 Homework 2

NLP Analysis on *Moby-Dick*

Adjective and Adverb Phrases

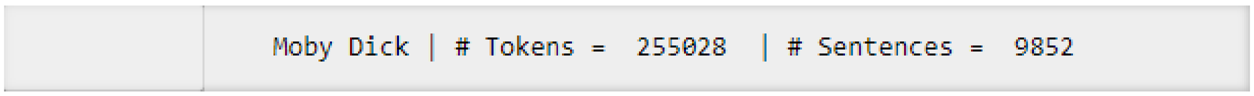
IST 664 Syracuse University

November 2022

In my previous assignment, I analyzed n-gram frequencies in the books *Moby-Dick* by Herman Melville and *Hamlet* by William Shakespeare. Now, in this assignment, I will be building on that work by analyzing adjective and adverb phrases in *Moby-Dick*.

I will start off by loading the text from *Moby-Dick* via the Gutenberg corpus in the nltk package and will display the number of words and sentences contained in the text.

```
# Import the text from Moby Dick and Hamlet
text_mobydick = nltk.corpus.gutenberg.raw("melville-moby_dick.txt")

# Split the text into tokens
tokens_mobydick = nltk.word_tokenize(text_mobydick)

# Split the text into sentences
sent_mobydick = nltk.sent_tokenize(text_mobydick)

# Print how many words and sentences there are
print("Moby Dick | # Tokens = ",
      len(tokens_mobydick),
      " | # Sentences = ",
      len(sent_mobydick))
```



```
Moby Dick | # Tokens =  255028  | # Sentences =  9852
```

The number of tokens in *Moby-Dick* is 255,028 and the number of sentences is 9,852 sentences. The average length of a sentence runs around 26 tokens.

Next, I will be extracting adjective and adverb phrases. I will do this by using part of speech tagging and using a regex parser. For this task, I am keeping the grammar rules of an adjective phrase and an adverb phrase simple.

An adjective phrase is defined as either:

(a) an adjective followed by a noun

(b) a noun followed by an adjective

An adverb phrase is defined as either:

(a) an adverb followed by a verb

(b) an adverb followed by a noun

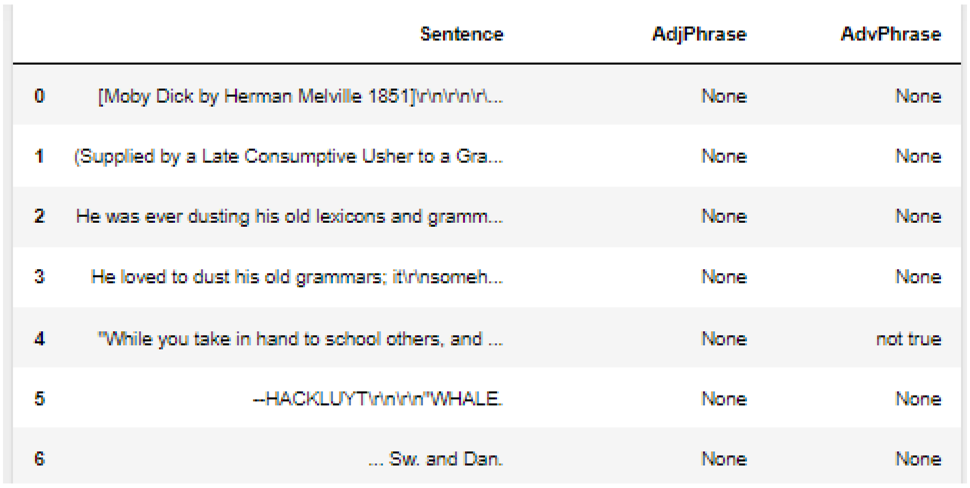
(c) an adverb followed by an adjective

The following code iterates through every sentence and determines if it contains an adjective phrase or an adverb phrase. If it does, the phrase will be extracted. The resulting information will be stored in a dataframe for further analysis.

```
# Define the grammar for an adjective phrase
adjp_grammar = """
ADJP: {<JJ><NN>}
      {<NN><JJ>}
"""

# Define the grammar for an adverb phrase
advp_grammar = """
ADVP: {<RB><V>}
      {<RB><NN>}
      {<RB><JJ>}
"""
```

```
# Define a function for retrieving phrases
def get_phrase(sentence, pattern, phrase):
    tokens = nltk.word_tokenize(sentence)
    tags = nltk.pos_tag(tokens)
    parser = nltk.RegexpParser(pattern)
    tree = parser.parse(tags)
    for sub1 in tree.subtrees():
        for sub2 in sub1:
            if type(sub2) != tuple:
                extract = ""
                for sub3 in sub2.leaves():
                    extract += " " + str(sub3[0])
                return extract

# Iterate through the sentences and retrieve the adj phrases
adj_phrases = []
adv_phrases = []
for sentence in sent_mobydick:
    try:
        adj_phrase = get_phrase(sentence, adjp_grammar, "ADJP")
        adv_phrase = get_phrase(sentence, advp_grammar, "ADVP")
        adj_phrases.append(adj_phrase)
        adv_phrases.append(adv_phrase)
    except:
        adj_phrases.append(None)
        adv_phrases.append(None)

# Put together a dataframe of the sentence, phrase, and test
mobydick_df = pd.DataFrame(list(zip(
    sent_mobydick,
    adj_phrases,
    adv_phrases)),
    columns = ["Sentence", "AdjPhrase", "AdvPhrase"]
)
```

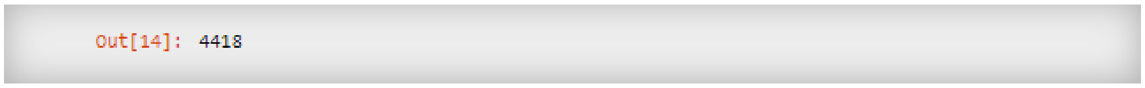
Here is what the dataframe looks like. Note, if the "AdjPhrase" or "AdvPhrase" column shows a value of "None", that means that there was no corresponding phrase found.



	Sentence	AdjPhrase	AdvPhrase
0	[Moby Dick by Herman Melville 1851]\r\n\r\n\r\...	None	None
1	(Supplied by a Late Consumptive Usher to a Gra...	None	None
2	He was ever dusting his old lexicons and gramm...	None	None
3	He loved to dust his old grammars; it\r\nsomeh...	None	None
4	"While you take in hand to school others, and ...	None	not true
5	--HACKLUYT\r\n\r\n"WHALE.	None	None
6	... Sw. and Dan.	None	None

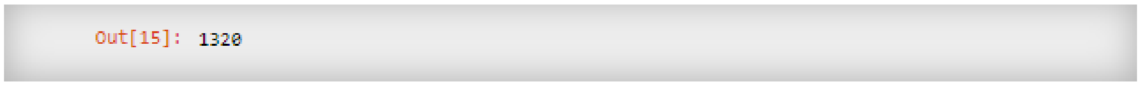
The sample of the dataframe above does not tell the full story because most of the values are showing as "None". The point is just to show what the dataframe looks like. So how many adjective phrases and adverb phrases were found?

```
# How many sentences with adjective phrases were there?
sum([phrase != None for phrase in list(mobydick_df["AdjPhrase"])])
```



```
Out[14]: 4418
```

```
# How many sentences with adverb phrases were there?
sum([phrase != None for phrase in list(mobydick_df["AdvPhrase"])])
```



```
Out[15]: 1320
```

There were 4,418 sentences that had an adjective phrase (44.8%) and 1,320 sentences that had an adverb phrase (13.4%).

Next, I will show summary statistics and histograms of the length of sentences. I will show this information this for four different groupings depending on whether an adjective phrase or an adverb phrase was found:

(a) Sentences that have an adjective phrase

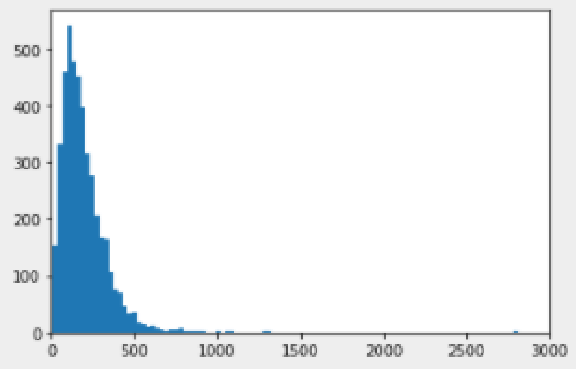
(b) Sentences that do not have adjective phrase

(c) Sentences that have an adverb phrase

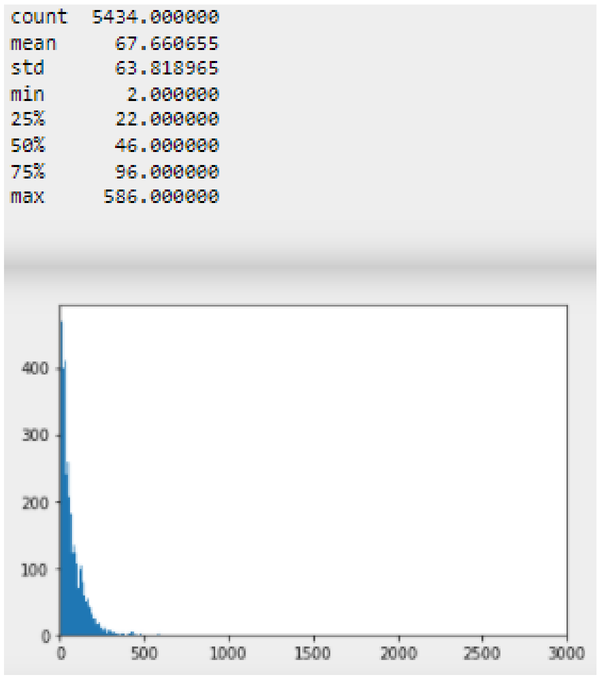
(d) Sentences that do not have an adverb phrase

Lengths of sentences that have an adjective phrase

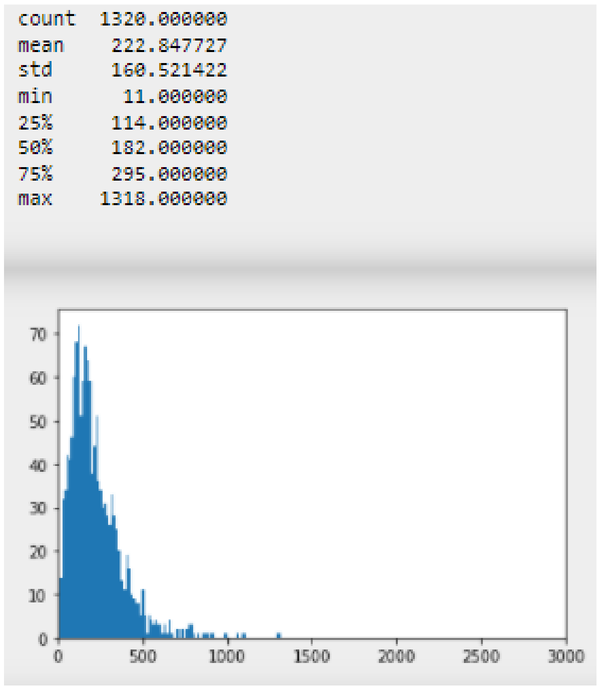
```
count  4418.00000
mean    192.52603
std     136.89557
min       9.00000
25%     101.00000
50%     163.00000
75%     249.00000
max    2809.00000
```



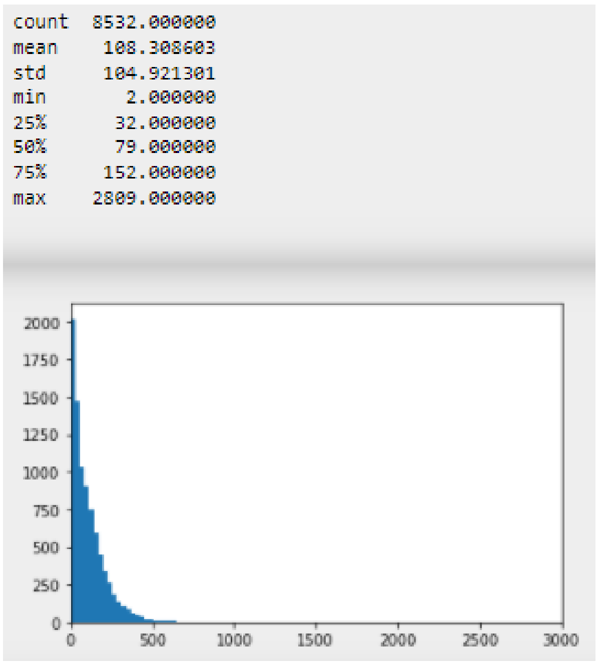
Lengths of sentences that do not have an adjective phrase


```
count   5434.000000
mean      67.660655
std       63.818965
min        2.000000
25%       22.000000
50%       46.000000
75%       96.000000
max      586.000000
```

Lengths of sentences that have an adverb phrase


```
count   1320.000000
mean     222.847727
std      160.521422
min       11.000000
25%      114.000000
50%      182.000000
75%      295.000000
max     1318.000000
```

Lengths of sentences that do not have an adverb phrase



```
count   8532.000000
mean     108.308603
std      104.921301
min        2.000000
25%       32.000000
50%       79.000000
75%      152.000000
max     2809.000000
```

Lastly, I will create a frequency distribution of the adjective phrases and adverb phrases and display the top 50 most common phrases.

Top 50 adjective phrases

```
# Create a frequency distribution of adjective phrases and print out the top 50
fdist = nltk.FreqDist(list(mobydick_df[mobydick_df["AdjPhrase"].notnull()].AdjPhrase))
for pair in fdist.most_common(50): print(pair)
```

```
(' old man', 53)
(' sperm whale', 27)
(' white whale', 17)
(' young man', 10)
(' chief mate', 10)
(' first place', 9)
(' first time', 9)
(' same time', 8)
(' other side', 8)
(' other way', 7)
(' present day', 7)
(' right whale', 7)
(' good time', 7)
(" old man's", 6)
(' same instant', 6)
(' poor fellow', 6)
(' general thing', 6)
(' same way', 5)
(' long time', 5)
(' thou art', 5)
(' last night', 5)
(' whale fishery', 5)
(' mortal man', 5)
(' dead whale', 4)
(' great whale', 4)
(' whaling voyage', 4)
(' next morning', 4)
(' small degree', 4)
(' old woman', 4)
(' Next morning', 4)
(' next day', 4)
(' second mate', 4)
(' white water', 4)
(' other whale', 4)
(' upper part', 4)
(' present case', 4)
(' stricken whale', 4)
(' poor devil', 3)
(' last day', 3)
(' young fellow', 3)
(' full ship', 3)
(' little plan', 3)
(' good deal', 3)
(' high time', 3)
(' next moment', 3)
(' past night', 3)
(' first glimpse', 3)
(' poor lad', 3)
(' present time', 3)
(' proper place', 3)
```

Top 50 adverb phrases

```
# Create a frequency distribution of adverb phrases and print out the top 50
fdist = nltk.FreqDist(list(mobydick_df[mobydick_df["AdvPhrase"].notnull()].AdvPhrase))
for pair in fdist.most_common(50): print(pair)
```

```
(' so much', 35)
(' so many', 14)
(' too much', 10)
(' as much', 8)
(' never mind', 7)
(' very large', 6)
(' so nigh', 6)
(' too late', 4)
(' so wide', 4)
(' so good', 4)
(' very little', 4)
(' as good', 4)
(' so saying', 4)
(' not much', 4)
(' very many', 4)
(' very curious', 4)
(' very much', 4)
(' so great', 3)
(' still stranger', 3)
(' very similar', 3)
(' far distant', 3)
(' ever such', 3)
(' Very good', 3)
(' so remarkable', 3)
(' so small', 3)
(' so strange', 3)
(' so easy', 3)
(' very act', 3)
(' very few', 3)
(' far other', 3)
(' very learned', 3)
(' not true', 2)
(' very great', 2)
(' ever heard', 2)
(' still further', 2)
(' too many', 2)
(' very reason', 2)
(' as cool', 2)
(' never heard', 2)
(' somewhat similar', 2)
(' only alive', 2)
(' so outlandish', 2)
(' very old', 2)
(' so intense', 2)
(' so prolonged', 2)
(' very convenient', 2)
(' very point', 2)
(' not something', 2)
(' well nigh', 2)
(' uncommonly heedful', 2)
```

Conclusions

I noticed that in sentences that contained an adjective phrase or an adverb phrase, the length of the sentence tends to be longer. I think this makes sense because these types of sentences form the substance of the story. For example, typically there is more context around these phrases, and the author is using adjectives and adverbs to make points of emphasis.

I was surprised to see that there were a lot of sentences that did not contain an adjective phrase or an adverb phrase, but I attribute this to limitations in my methodology. I had a difficult time finding resources for extracting phrases, so I had to create code from scratch. Unfortunately, I was only successful at extracting one phrase out of a sentence, so if there were multiple phrases in one sentence I did not capture that.

There are several ways that sentiment analysis could be conducted on this body of text. While one way to do it would be to score individual words, I think a better way would be to score sentences. If only words are scored, then there may be some context that gets missed. For example, negations are often missed by sentiment scorers. For some tasks, scoring by words might be sufficient but for a book I believe that scoring at a higher level would provide better results.