Name: Sai Sandeep Varma Mudundi
Gnumber: G01352322

Name: Rajeev Priyatam Panchadula
Gnumber: G01333080

Assignment-1

Introduction

This assignment consists of using a PySpark program to calculate word count, the average and standard deviation of a word in a window of 4 years, the frequency of co-occurrence of words, and the lift between two words. The data being used is the text transcriptions of the State of the Union Addresses given by Presidents to Congress from 1790 to the present year. The data consists of files named yyyymmdd.txt, where yyyy is the year, mm is the month, and dd is the day on which the address was given.

Data Gathering:

To get the data we used a simple web scrapping program template to load HTML files from the SOTU website and store it in the local directory.

Part 1:

Preprocessing:

To clean the raw data from the State of the Union Website the below-preprocessing steps:

1. Initially we remove all the tags before the h3 tag. Later we removed the footer of the HTML file. This gives all the text content within the paragraph tags
2. HTML tags with a '< />' pattern are removed
3. New line and tab characters are removed
4. All punctuation and extra white spaces are removed
5. All numeric digit occurrences are removed
6. Lastly, we use the nltk library to get a list of stop words, then we replace each stop word with a space in the speech data.

We used two different stop word removing functions. One is used to clean stop words for data required in part 1. In part 2 we need to split by sentences to get the frequency of co-occurrence of words in the SOTU text within the same sentence. So, we have not removed the full stop /'.' Punctuation.

Computing the average and standard deviation of times a word appears in a window of four years:

Initially, to get the total average, we split the text data to get the word list. Then we initialized 1 to each word occurrence. Then we used '*reduceByKey(lambda x,y: x+y)*' function to get the frequency count of a particular word. Then a year range was made for every 4 years, and the words were associated with this year range id(i.e., 2009 to 2012). Then the average occurrence of a word for these windows was calculated. To get the standard deviation we have used the python one-liner ‘*(sum((x-avg)**2 for x in lst) / len(lst))**0.5*’ to replicate(the square root of the below formula gives standard deviation):

$$S^2 = \frac{\sum(x_i - \bar{x})^2}{n-1}$$

S^2 = sample variance
x_i = the value of the one observation
$\bar{x}$ = the mean value of all observations
n = the number of observations

Sample Output:

```
The word madame in the window of: 2009 to 2012 appears with an average of 0.5 and standard deviation(during 2009 to 2012
window ) is 0.5443310539518174

The word lady in the window of: 2021 to 2024 appears with an average of 0.5 and standard deviation(during 2021 to 2024 window
) is 2.457038265277331

The word united in the window of: 2009 to 2012 appears with an average of 5.5 and standard deviation(during 2009 to 2012
window ) is 26.606877417801694

The word united in the window of: 2013 to 2016 appears with an average of 6.25 and standard deviation(during 2013 to 2016
window ) is 26.606877417801694

The word united in the window of: 2017 to 2020 appears with an average of 15.75 and standard deviation(during 2017 to 2020
window ) is 26.606877417801694

The word economy in the window of: 2021 to 2024 appears with an average of 1.75 and standard deviation(during 2021 to 2024
window ) is 29.66354540384419
```

Getting words that appear in the year following the window with a frequency that exceeds the average plus two standard deviations:

Using the previous result of average and standard deviation, for a particular word we have calculated the average plus 2 times standard deviation threshold value as an RDD column. Then for a particular word in the following year past the standard 4-year window, we checked if the frequency of a particular word exceeds the threshold or not. Then we filter and get those words. This activity could be useful to see and check patterns in a speech followed by a presidential term. This also helps to know if the next presidential candidate used the same set of words or was in opposition to the words from the previous 4-year window. Below is the sample output of words that fall under this category.

```
Consequences    task
plunged         periods
holding         certainly
talents         computers
perfectly       invested
demand          wasted
earliest        developing
bullet          institute
```

Name: Sai Sandeep Varma Mudundi
Gnumber: G01352322

Name: Rajeev Priyatam Panchadula
Gnumber: G01333080

Part – 2

Counting the frequency of co-occurrence of words in the SOTU text within the same sentence:

Initially, we created a new spark session for part 2. We have used the 'SparkSession' library to get it started. The dataset used here is the same as part 1, but in this case, we have considered the entire data of all years from 1780-present. After preprocessing the dataset by removing stop words, HTML commands, and other punctuations(For this part we have retained the 'full stop' punctuation as we need it to split the speech content based on sentences). To achieve this initially we removed all the content other than paragraph tags and cleaned the data using a similar method as in the part 1 section. We have used regex to ignore '.' Punctuation and writing a similar function to clean data so that it can be used in this section.

We have taken the template data frames as a reference and built the functionality around them. After Splitting the data based on sentences, we have used 2 copies of the data frame that in turn are being used as a part of the join statement. This gives us different combinations of two pairs. Then we counted the frequency of the occurrence of these pairs and filtered out the word pairs that do not appear more than 10 times.

Here is a sample **Output with 20 frequent pairs of words (not necessarily the top 20)** whose frequency of co-occurrence within the same sentence is more than 10 times:

Word -1	**Word-2**	**Frequency**
case	shall	11
shall	taken	11
conservation	natural	12
goods	value	12
federal	spending	12
daily	would	12
officers	services	12
condition	states	12
nations	tribes	12
number	supreme	12
direct	would	12
cleavage	men	12

Name: Sai Sandeep Varma Mudundi
Gnumber: G01352322

Name: Rajeev Priyatam Panchadula
Gnumber: G01333080

judges	let	12
district	prohibition	12
government	may	12
islands	united	12
circuit	courts	25
federal	government	27
circulating	notes	35
energy	production	40
service	veterans	40
civil	service	42
ship	submarines	54

Computing the Lift between two words:

Lift is a very interesting technique that shows how words are associated with each other.

It is calculated using :

Lift(AB) = P(AB)/(P(A)*P(B))

Where P(AB)= probability of the pair AB = frequency(AB)/N_sentences

P(A) = probability of word A = frequency(A)/N_sentences

P(B) = probability of word B = frequency(B)/N_sentences

N_sentences = number of sentences found in ALL the speeches.
Reference from https://piazza.com/class/l74xcqzm8nw1rj/post/32

The Lift measures the probability of A and B occurring together divided by the probability of A and B occurring if they were independent events. If A and B are independent, then the Lift == 1. If they occur together more often than if they were independent, then Lift > 1. We have implemented the above formulas on the columns in respective data frames to get the lift between a pair of words. Words whose lift is greater than 3 are considered as a part of the output. The co-occurrence of a pair of words with a lift value of more than 3 would mean that these 2 words have the greatest association within the presidential speeches. Having a lift below 3 would generally mean that those 2 worlds are non-related. This helps us analyze the speeches to see reoccurring patterns and predict the word pairs to be repeated in the next speech with good confidence.
Below is a sample output with two words and the lift associated with them.

Name: Sai Sandeep Varma Mudundi
Gnumber: G01352322

Name: Rajeev Priyatam Panchadula
Gnumber: G01333080

Word A	**Word B**	**Lift**
abandon	bank	4.30042
breaking	communication	20
gross	injustice	20
imperial	wrongs	19.99674
gratifying	hospitality	12.83654
founded	reparation	12.83654
authorized	captures	9.287659
feeling	threatening	9.287659
agreement	proclaimed	7.86501
extradition	threatened	7.862996
gulf	supervision	7.074885
happiness	territory	4.095638
agriculture	report	3.612353

Conclusion

Overall, the assignment was a good exercise in knowing how to scale, reduce and perform data mining analytic techniques to find patterns associated with words. The lift associated with words in part 2 showed patterns of frequently associated words used by presidents. Generally, candidates used words like ‘gross injustice’, and ‘agriculture report’ together to talk about a social issue or government-issued report/survey.

References:

1. https://blog.finxter.com/how-to-get-the-standard-deviation-of-a-python-list/
2. https://sparkbyexamples.com/
3. https://cs.gmu.edu/~dbarbara/CS657/df_pairs.txt
4. https://piazza.com/class/l74xcqzm8nw1rj/post/32