BIG Data – Spark

<u>Table of Content</u>

sources of information:

https://www.geeksforgeeks.org/introduction-pyspark-distributed-computing-apache-/spark

https://spark.apache.org/docs/0.6.2/api/core/spark/RDD.html

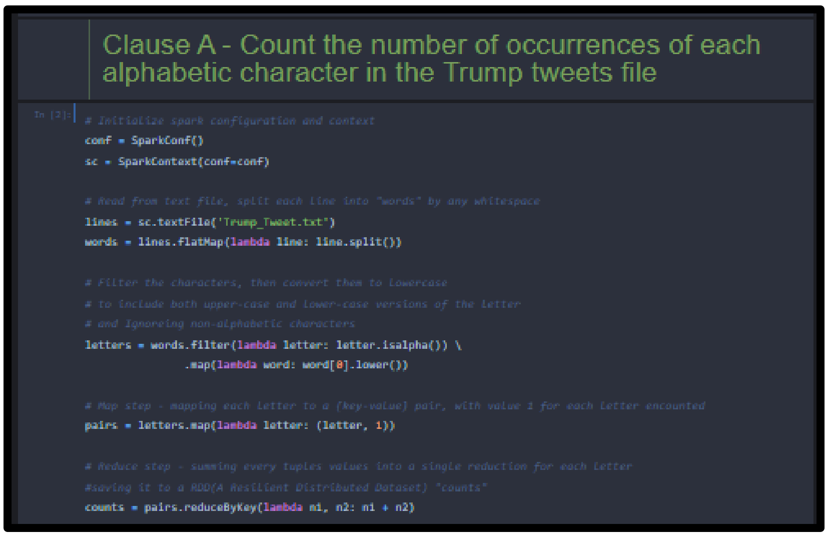
Lecture presentation.

<u>Terms definition</u>

Terms	**Definition**
Spark	Apache Spark is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming clusters with implicit data parallelism and fault tolerance
NLTK	The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing for English written in the Python programming language
RDD	They are a distributed collection of objects, which are stored in memory or on disks of different machines of a cluster. A single RDD can be divided into multiple logical partitions so that these partitions can be stored and processed on different machines of a cluster.
N-Grams	Is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application.
Stop Words	Stop words are the words in a stop list which are filtered out before or after processing of natural language data because they are insignificant.

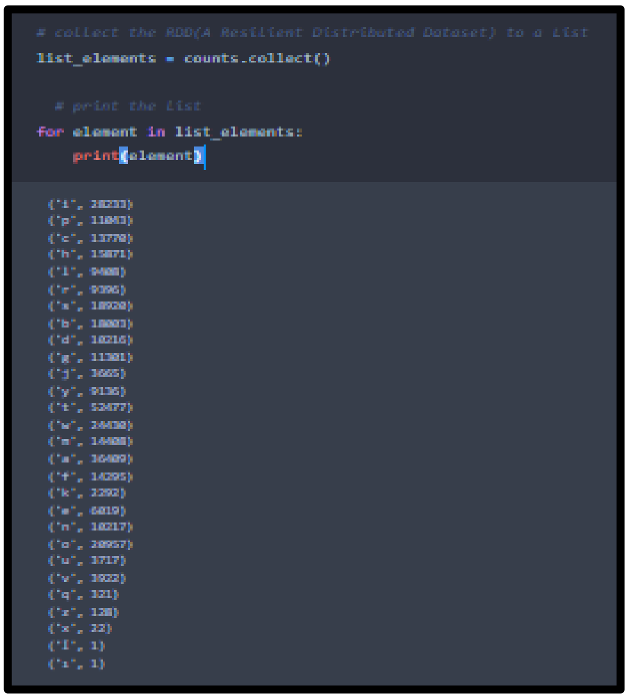
Count letters:

Count the number of occurrences of each alphabetic character in a file. The count for each letter should be case-insensitive. Ignore non-alphabetic characters. (20 points).



Clause A - Count the number of occurrences of each alphabetic character in the Trump tweets file

```
In [2]:
# Initialize spark configuration and context
conf = SparkConf()
sc = SparkContext(conf=conf)

# Read from text file, split each line into "words" by any whitespace
lines = sc.textFile('Trump_Tweet.txt")
words = lines.flatMap(lambda line: line.split())

# Filter the characters, then convert them to lowercase
# to include both upper-case and lower-case versions of the letter
# and Ignoreing non-alphabetic characters
letters = words.filter(lambda letter: letter.isalpha()) \
               .map(lambda word: word[0].lower())

# Map step - mapping each letter to a (key-value) pair, with value 1 for each letter encounted
pairs = letters.map(lambda letter: (letter, 1))

# Reduce step - summing every tuples values into a single reduction for each letter
#saving it to a RDD(A Resilient Distributed Dataset) "counts"
counts = pairs.reduceByKey(lambda n1, n2: n1 + n2)
```

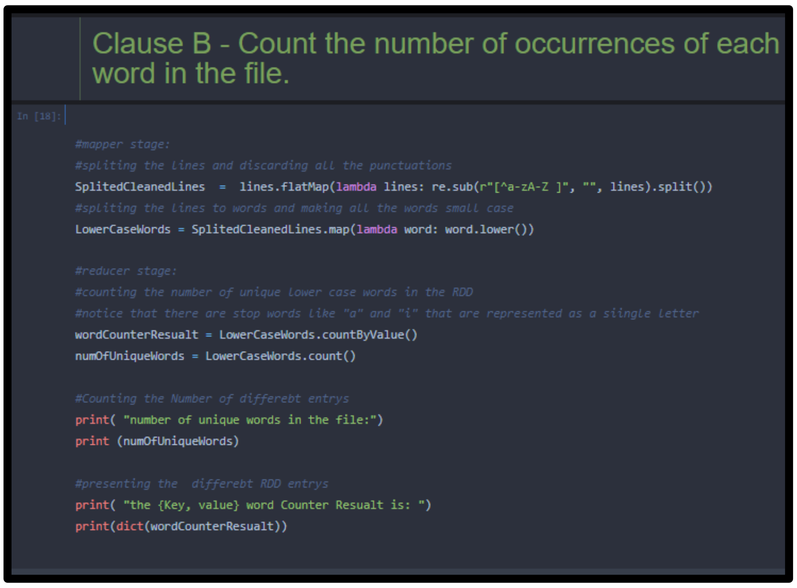


```
# collect the RDD(A Resilient Distributed Dataset) to a list
list_elements = counts.collect()

  # print the list
for element in list_elements:
    print(element)
```

```
('i', 28233)
('p', 11043)
('c', 13770)
('h', 15871)
('l', 9488)
('r', 9396)
('s', 18928)
('b', 10883)
('d', 10216)
('g', 11301)
('j', 3665)
('y', 9136)
('t', 52477)
('w', 24430)
('m', 14408)
('a', 36409)
('f', 14295)
('k', 2292)
('e', 6819)
('n', 10217)
('o', 20957)
('u', 3717)
('v', 3922)
('q', 321)
('z', 128)
('x', 22)
('I', 1)
('ı', 1)
```

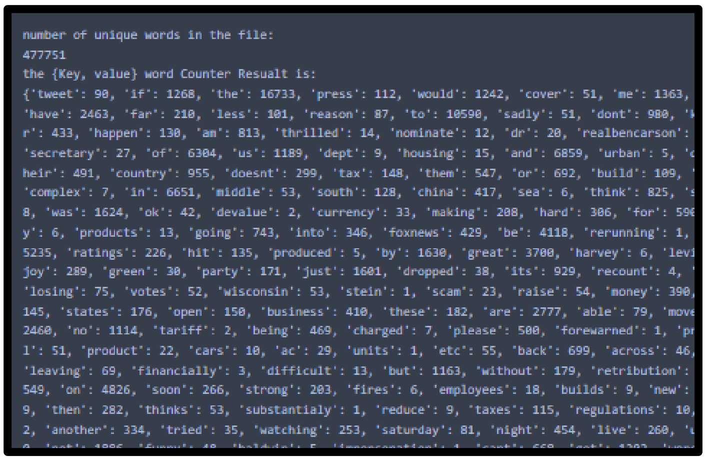
Count Words:

Count the number of occurrences of each word in the file. The count should be case-insensitive and ignore punctuations. (20 points).



Clause B - Count the number of occurrences of each word in the file.

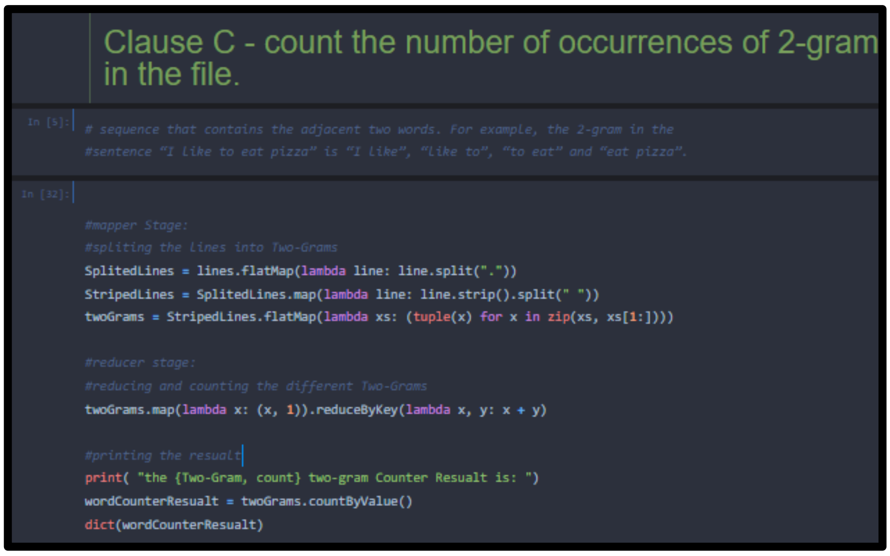
```
In [18]:

#mapper stage:
#spliting the lines and discarding all the punctuations
SplitedCleanedLines  =  lines.flatMap(lambda lines: re.sub(r"[^a-zA-Z ]", "", lines).split())
#spliting the lines to words and making all the words small case
LowerCaseWords = SplitedCleanedLines.map(lambda word: word.lower())

#reducer stage:
#counting the number of unique lower case words in the RDD
#notice that there are stop words like "a" and "i" that are represented as a siingle letter
wordCounterResualt = LowerCaseWords.countByValue()
numOfUniqueWords = LowerCaseWords.count()

#Counting the Number of differebt entrys
print( "number of unique words in the file:")
print (numOfUniqueWords)

#presenting the  differebt RDD entrys
print( "the {Key, value} word Counter Resualt is: ")
print(dict(wordCounterResualt))
```



```
number of unique words in the file:
477751
the {Key, value} word Counter Resualt is:
{'tweet': 90, 'if': 1268, 'the': 16733, 'press': 112, 'would': 1242, 'cover': 51, 'me': 1363,
'have': 2463, 'far': 210, 'less': 101, 'reason': 87, 'to': 10590, 'sadly': 51, 'dont': 980, 'k
r': 433, 'happen': 130, 'am': 813, 'thrilled': 14, 'nominate': 12, 'dr': 20, 'realbencarson':
'secretary': 27, 'of': 6304, 'us': 1189, 'dept': 9, 'housing': 15, 'and': 6859, 'urban': 5, 'c
heir': 491, 'country': 955, 'doesnt': 299, 'tax': 148, 'them': 547, 'or': 692, 'build': 109, '
'complex': 7, 'in': 6651, 'middle': 53, 'south': 128, 'china': 417, 'sea': 6, 'think': 825, 's
8, 'was': 1624, 'ok': 42, 'devalue': 2, 'currency': 33, 'making': 208, 'hard': 306, 'for': 59€
y': 6, 'products': 13, 'going': 743, 'into': 346, 'foxnews': 429, 'be': 4118, 'rerunning': 1,
5235, 'ratings': 226, 'hit': 135, 'produced': 5, 'by': 1630, 'great': 3700, 'harvey': 6, 'levi
joy': 289, 'green': 30, 'party': 171, 'just': 1601, 'dropped': 38, 'its': 929, 'recount': 4, '
'losing': 75, 'votes': 52, 'wisconsin': 53, 'stein': 1, 'scam': 23, 'raise': 54, 'money': 390,
145, 'states': 176, 'open': 150, 'business': 410, 'these': 182, 'are': 2777, 'able': 79, 'move
2460, 'no': 1114, 'tariff': 2, 'being': 469, 'charged': 7, 'please': 500, 'forewarned': 1, 'pr
1': 51, 'product': 22, 'cars': 10, 'ac': 29, 'units': 1, 'etc': 55, 'back': 699, 'across': 46,
'leaving': 69, 'financially': 3, 'difficult': 13, 'but': 1163, 'without': 179, 'retribution':
549, 'on': 4826, 'soon': 266, 'strong': 203, 'fires': 6, 'employees': 18, 'builds': 9, 'new':
9, 'then': 282, 'thinks': 53, 'substantialy': 1, 'reduce': 9, 'taxes': 115, 'regulations': 10,
2, 'another': 334, 'tried': 35, 'watching': 253, 'saturday': 81, 'night': 454, 'live': 260, 'u
```

Count 2-grams:

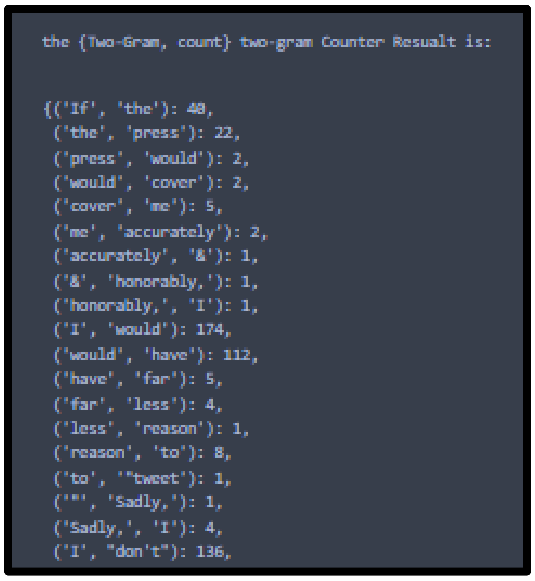
Based on b, count the number of occurrences of 2-gram in the file. A 2-gram is a sequence that contains the adjacent two words. For example, the 2-gram in the sentence “I like to eat pizza” is “I like”, “like to”, “to eat” and “eat pizza”. (20 points).



Clause C - count the number of occurrences of 2-gram in the file.

```
In [5]:
# sequence that contains the adjacent two words. For example, the 2-gram in the
#sentence “I like to eat pizza” is “I like”, “like to”, “to eat” and “eat pizza”.
```

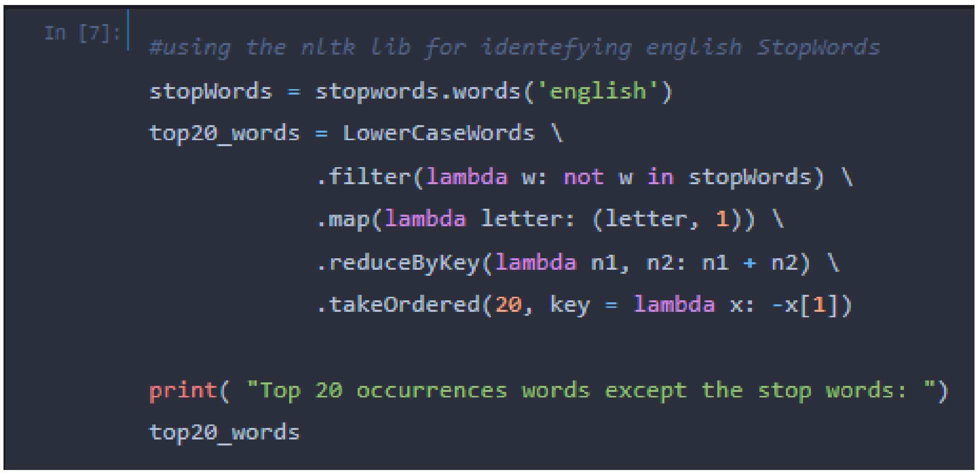
```
In [32]:

#mapper Stage:
#spliting the lines into Two-Grams
SplitedLines = lines.flatMap(lambda line: line.split("."))
StripedLines = SplitedLines.map(lambda line: line.strip().split(" "))
twoGrams = StripedLines.flatMap(lambda xs: (tuple(x) for x in zip(xs, xs[1:])))

#reducer stage:
#reducing and counting the different Two-Grams
twoGrams.map(lambda x: (x, 1)).reduceByKey(lambda x, y: x + y)

#printing the resualt
print( "the {Two-Gram, count} two-gram Counter Resualt is: ")
wordCounterResualt = twoGrams.countByValue()
dict(wordCounterResualt)
```



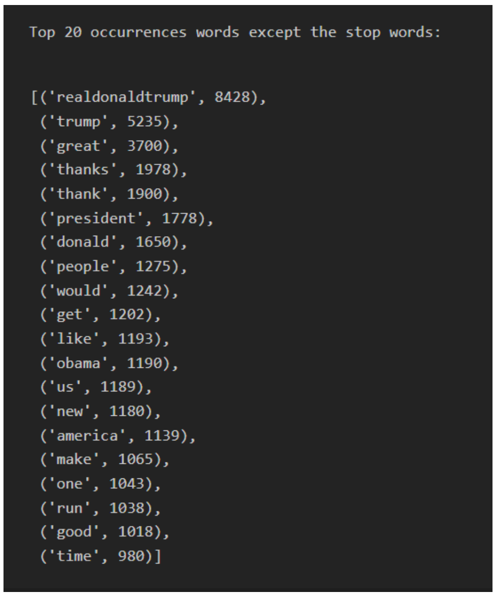
```
the {Two-Gram, count} two-gram Counter Resualt is:


{('If', 'the'): 48,
 ('the', 'press'): 22,
 ('press', 'would'): 2,
 ('would', 'cover'): 2,
 ('cover', 'me'): 5,
 ('me', 'accurately'): 2,
 ('accurately', '&'): 1,
 ('&', 'honorably,'): 1,
 ('honorably,', 'I'): 1,
 ('I', 'would'): 174,
 ('would', 'have'): 112,
 ('have', 'far'): 5,
 ('far', 'less'): 4,
 ('less', 'reason'): 1,
 ('reason', 'to'): 8,
 ('to', '"tweet'): 1,
 ('"', 'Sadly,'): 1,
 ('Sadly,', 'I'): 4,
 ('I', "don't"): 136,
```

Most Frequently words:

Based on b, obtain the top 20 most frequently occurred words except the stop words such as “a”, “the”, “this” and so on. The stop words can be imported by stop-words package (see https://pypi.org/project/stop-words/) (30 marks).



```
In [7]:
#using the nltk lib for identefying english StopWords
stopWords = stopwords.words('english')
top20_words = LowerCaseWords \
            .filter(lambda w: not w in stopWords) \
            .map(lambda letter: (letter, 1)) \
            .reduceByKey(lambda n1, n2: n1 + n2) \
            .takeOrdered(20, key = lambda x: -x[1])

print( "Top 20 occurrences words except the stop words: ")
top20_words
```



```
Top 20 occurrences words except the stop words:


[('realdonaldtrump', 8428),
 ('trump', 5235),
 ('great', 3700),
 ('thanks', 1978),
 ('thank', 1900),
 ('president', 1778),
 ('donald', 1650),
 ('people', 1275),
 ('would', 1242),
 ('get', 1202),
 ('like', 1193),
 ('obama', 1190),
 ('us', 1189),
 ('new', 1180),
 ('america', 1139),
 ('make', 1065),
 ('one', 1043),
 ('run', 1038),
 ('good', 1018),
 ('time', 980)]
```

.