Hutch code

I did not run the pays-ark with Notebook - I ran it 9 times as a chunk on the server(7 for classics, 2 or internet)

Since I ran the code on the server and have been learning R for plotting all semester, I did my plots in R

Below are the chunks that I used:

```
# Code for Classics


#set context
from pyspark import SparkContext, SparkConf
conf =
SparkConf().setAppName("miniProject").setMaster("local[
*]")
sc = SparkContext.getOrCreate(conf)

#packages
import numpy as np
import scipy.sparse as sps
import re

#read file
file_name = "/spring2021/project1/NAME.csv"

myRDD = sc.textFile(file_name)



def pullout(x):
        collist = x.split(",")
        if (len(collist) < 9):
            return ","
```

```
        return collist[8]



myRDDList = myRDD.map(lambda x : pullout(x))

#cleaning function - removes punctuation
def wordclean(x):
    return re.sub("[^a-zA-Z\s]+","", x).lower().strip()

#new word list sans punc
myRDDList = myRDDList.map(lambda x : wordclean(x))

#remove spaces
myRDDwords = myRDDList.flatMap( lambda x: x.split(" "))
myRDDwords = myRDDwords.filter(lambda x: (len(x) != 0)
and (len(x) < 22))

#create tuple - (word, length of word)
myRDDwordPairsLen =  myRDDwords.map(lambda x:
(x,len(x)))


#extract values
myRddwordPairLenValues = myRDDwordPairsLen.values()

#compute average - list with stop words
lensum = myRddwordPairLenValues.sum()
lencount = myRddwordPairLenValues.count()
avlen = lensum/lencount
avlen

mylenpair =  myRDDwords.map(lambda x: (len(x),1))

mywordlenssum = mylenpair.reduceByKey(lambda a,b: a+b)
#(lenght,count)
```

```
#nltk imports for stop word removal
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords

stopwords = stopwords.words('english')

#filter out stop words
myrddNS = myRDDwords.filter(lambda x : x[0] not in
stopwords)

#create tuple - (word, length of word)
myrddNSPairsLen =  myrddNS.map(lambda x: (x,len(x)))

#compute average word length - list w/o stop words

myrddNSPairsLenValues = myrddNSPairsLen.values()


lensumNS = myrddNSPairsLenValues.sum()

lencountNS = myrddNSPairsLenValues.count()

avNS = lensumNS / lencountNS


mylenpairNS =  myrddNS.map(lambda x: (len(x),1))

mywordlenssumNS = mylenpairNS.reduceByKey(lambda a,b:
a+b)  #(lenght,count)


lensum
lencount
avlen
```

```
mywordlenssum.take(30)


lensumNS
lencountNS
avNS
mywordlenssumNS.take(30)




myRDDwords = myRDDList.filter(lambda x: len(x) != 0)
myRDDwords = myRDDlist.filter(lambda x: len(x) != 0)
```

Code for blogtext

```
#set context
from pyspark import SparkContext, SparkConf
conf =
SparkConf().setAppName("miniProject").setMaster("local[
*]")
sc = SparkContext.getOrCreate(conf)

#packages
import numpy as np
import scipy.sparse as sps
import re

#read file
file_name = "/spring2021/project1/blogtext.csv"

myRDD = sc.textFile(file_name)
```

```
#remove correct (last) column    ignore rows with fewer
columns
def pullout(x):
        collist = x.split(",")
        if (len(collist) < 9):
            return ","
        return collist[8]

myRDDList = myRDD.map(lambda x : pullout(x))

#cleaning function - removes punctuation and numbers
def wordclean(x):
    return re.sub("[^a-zA-Z\s]+","", x).lower().strip()

#new word list sans punc
myRDDList = myRDDList.map(lambda x : wordclean(x))

#remove spaces
myRDDwords = myRDDList.flatMap( lambda x: x.split(" "))
myRDDwords = myRDDwords.filter(lambda x: (len(x) != 0)
and (len(x) < 22))

#create tuple - (word, length of word)
myRDDwordPairsLen =  myRDDwords.map(lambda x:
(x,len(x)))


#extract values
myRddwordPairLenValues = myRDDwordPairsLen.values()

#compute average - list with stop words
lensum = myRddwordPairLenValues.sum()
lencount = myRddwordPairLenValues.count()
avlen = lensum/lencount
avlen

mylenpair =  myRDDwords.map(lambda x: (len(x),1))
```

```
mywordlenssum = mylenpair.reduceByKey(lambda a,b: a+b)
#(lenght,count)



#nltk imports for stop word removal
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords

stopwords = stopwords.words('english')

#filter out stop words
myrddNS = myRDDwords.filter(lambda x : x[0] not in
stopwords)

#create tuple - (word, length of word)
myrddNSPairsLen =  myrddNS.map(lambda x: (x,len(x)))

#compute average word length - list w/o stop words

myrddNSPairsLenValues = myrddNSPairsLen.values()


lensumNS = myrddNSPairsLenValues.sum()

lencountNS = myrddNSPairsLenValues.count()

avNS = lensumNS / lencountNS


mylenpairNS =  myrddNS.map(lambda x: (len(x),1))

mywordlenssumNS = mylenpairNS.reduceByKey(lambda a,b:
a+b)  #(lenght,count)
```

```
lensum
lencount
avlen
mywordlenssum.take(30)


lensumNS
lencountNS
avNS
mywordlenssumNS.take(30)




#myRDDwords = myRDDList.filter(lambda x: len(x) != 0)
#myRDDwords = myRDDlist.filter(lambda x: len(x) != 0)
```

code for hacker

```
***df = spark.read.csv("/spring2021/project1/
hacker_news_sample.csv", header=True, inferSchema=True)

myRDD = df.rdd

myRDD = myRDD.map(lambda x:x[2]).filter(lambda x: x is
not None)
***



#set context
from pyspark import SparkContext, SparkConf
conf =
SparkConf().setAppName("miniProject").setMaster("local[
*]")
sc = SparkContext.getOrCreate(conf)
```

```
#packages
import numpy as np
import scipy.sparse as sps
import re


%%%
df = spark.read.csv("/spring2021/project1/
hacker_news_sample.csv", header=True, inferSchema=True)

myRDD = df


#read file
#file_name = "/spring2021/project1/"hacker_news.csv"

#myRDD = sc.textFile(file_name)

#text column of Hacker News.
myRDD = myRDD.map(lambda x:x[2]).filter(lambda x: x is
not None)
%%%

#remove correct ????? column    ignore rows with fewer
columns
#def pullout(x):
#        collist = x.split(",")
    #    if (len(collist) < 9):
    #        return ","
    #    return collist

#myRDDList = myRDD.map(lambda x : pullout(x))
#myRDDList.take(2)

def removehtml(line):
    cleanr = re.compile('<.*?>|&([a-z0-9]+|#[0-9]{1,6}|
#x[0-9a-f]{1,6});')
    return re.sub(cleanr, '', line)
```

```
mmy_rdd = my_rdd.map(removehtml)

my_rdd = my_rdd.map(removePuncAlt)
my_rdd = my_rdd.map(removeNums)
#Hacker_Words_rdd = Hacker_rdd.flatMap(lambda
line:line.split())


myRDDList = myRDD
#cleaning function - removes punctuation and numbers
def wordclean(x):
    return re.sub("[^a-zA-Z\s]+","", x).lower().strip()

#new word list sans punc
#myRDDList = myRDDList.map(lambda x : wordclean(x))

myRDD = myRDDList.map(lambda x : wordclean(x))

#new word list sans punc
myRDDList = myRDD.map(lambda x : wordclean(x))

myRDDList = myRDDList.map(lambda x : wordclean(x))




#remove spaces
myRDDwords = myRDDList.flatMap( lambda x: x.split(" "))
myRDDwords = myRDDwords.filter(lambda x: (len(x) != 0)
and (len(x) < 22))

#create tuple - (word, length of word)
myRDDwordPairsLen =  myRDDwords.map(lambda x:
(x,len(x)))


#extract values
```

```
myRddwordPairLenValues = myRDDwordPairsLen.values()

#compute average - list with stop words
lensum = myRddwordPairLenValues.sum()
lencount = myRddwordPairLenValues.count()
avlen = lensum/lencount
avlen

mylenpair =  myRDDwords.map(lambda x: (len(x),1))

mywordlenssum = mylenpair.reduceByKey(lambda a,b: a+b)
#(lenght,count)




#nltk imports for stop word removal
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords

stopwords = stopwords.words('english')

#filter out stop words
myrddNS = myRDDwords.filter(lambda x : x[0] not in
stopwords)

#create tuple - (word, length of word)
myrddNSPairsLen =  myrddNS.map(lambda x: (x,len(x)))

#compute average word length - list w/o stop words

myrddNSPairsLenValues = myrddNSPairsLen.values()


lensumNS = myrddNSPairsLenValues.sum()

lencountNS = myrddNSPairsLenValues.count()
```

```
avNS = lensumNS / lencountNS


mylenpairNS =  myrddNS.map(lambda x: (len(x),1))

mywordlenssumNS = mylenpairNS.reduceByKey(lambda a,b:
a+b)  #(lenght,count)


lensum
lencount
avlen
mywordlenssum.take(30)


lensumNS
lencountNS
avNS
mywordlenssumNS.take(30)


#myRDDwords = myRDDList.filter(lambda x: len(x) != 0)
#myRDDwords = myRDDlist.filter(lambda x: len(x) != 0)
```