import nltk

from nltk.tokenize import sent\_tokenize

from nltk.tokenize import word\_tokenize

from nltk.corpus import sentiwordnet as swn

from nltk.corpus import stopwords

import re

o = open("/Users/Apple/Desktop/output.csv","w")

a = open('/Users/Apple/Desktop/text.txt','r').read()

stop = stopwords.words('english')

s=sent\_tokenize(a)

n = "No.of.words\tStopwords\tNoun\tVerb\tAdj\tPos\_pol\tNeg\_pol\tPol\_val\tLevel\n"

o.write(n)

for i in range(len(s)):

k = nltk.word\_tokenize(s[i])

#print(k)

no\_words = (len(k))

# Stopwards

c =0

for j in k:

p = str(j).lower()

if p in stop:

c += 1

stopword = c

# Noun

t = nltk.pos\_tag(k)

m = 0

for j1 in range(len(t)):

if re.search("', 'NN",str(t[j1])):

m =m + 1

no\_noun=m

# Verb

m1 = 0

for j2 in range(len(t)):

if re.search("', 'VB",str(t[j2])):

m1 += 1

no\_verb = m1

# Adjective

m2 = 0

for j3 in range(len(t)):

if re.search("', 'JJ",str(t[j3])):

m2 += 1

no\_adj = m2

# Positive and Negative words

w = word\_tokenize(s[i])

p\_c=0

n\_c=0

for j in range(len(w)):

sy=swn.senti\_synsets(w[j])

l=[]

for k in sy:

p1=k.pos\_score()

n1=k.neg\_score()

polarity=float(p1)-float(n1)

l.append(float(polarity))

l2 = sorted(l)

if((len(l2)>0) and (l2[0] >= 0.0)):

l\_max = max(l2)

elif((len(l2)>0) and (l2[0] < 0.0)):

l\_max = max(l2)

elif((len(l2)>0) and (l2[0] < 0.0)):

l\_max = l2[0]

else:

l\_max=0.0

if float(l\_max)>=0.0:

p\_c+=1

elif float(l\_max)<0.0:

n\_c+=1

#print(l\_max)

op = (str(no\_words)+ '\t' +str(stopword) + '\t' + str(no\_noun) + '\t' + str(no\_verb) + '\t' + str(no\_adj) + '\t' + str(p\_c) + '\t' + str(n\_c))

o.write(op)

o.write("\n")

o.close()

import pandas as pd

import numpy as np

from sklearn.svm import SVC

from sklearn import svm

df = pd.read\_csv('/Users/Apple/Desktop/output.csv', sep='\t')

df

from sklearn.model\_selection import train\_test\_split

x = df[['No.of.words','Stopwords','Noun','Verb','Adj','Pos\_pol','Neg\_pol','Pol\_val']]

y = df[['Level']]

from sklearn.model\_selection import train\_test\_split

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 5,random\_state=42)

model = GaussianNB()

model.fit(x\_train,y\_train)

print(lm.intercept\_)

lm.coef\_

predictions = lm.predict(x\_test)

predictions

from sklearn.metrics import classification\_report

print(classification\_report(y\_test,predictions))