# -\*- coding: utf-8 -\*-

"""

Created on Fri Apr 07 00:20:26 2017

@author: chaitali

"""

#import nltk

#import sys

#from wordcloud import WordCloud

#from nltk.stem import WordNet.Lemmatizer

import json

from textblob import TextBlob

import matplotlib.pyplot as plt

import string

#def SentimentAnalysis(filename):

Tweets=[]

with open('tweet\_stream\_trump\_01.JSON','r') as f:

Tweets=json.load(f)

Tweets1=[]

for tweet in Tweets:

Tweets1.append(tweet['text'])

#.encode('ascii','replace').replace('?',' '))

#Tweets1.append(tweet['text'].encode(utf-8).replace('?',' '))'''

#print Tweets1

Tweets=[]

with open('tweet\_stream\_H1B\_100\_general-H1Btweets.JSON','r') as f:

Tweets=json.load(f)

tweet\_sub=[]

sumpol=0.0

sumsub=0.0

avgpol=0.0

avgsub=0.0

sub\_list = []

pol\_list = []

#if 'trump' in Tweets:

for tweet in Tweets:

# if 'trump' in tweet:

#tb = TextBlob(s)

s\_sen1= tweet['text']

tb\_sen = TextBlob(s\_sen1)

sumpol=sumpol+tb\_sen.polarity

sumsub=sumsub+tb\_sen.subjectivity

sub\_list.append(tb\_sen.subjectivity)

pol\_list.append(tb\_sen.polarity)

tweet\_sub.append(tb\_sen.subjectivity)

#return Tweets

avgpol= sumpol/len(Tweets)

avgsub=sumsub/len(Tweets)

print('avgpol=',avgpol)

print('avgsub=',avgsub)

print('sentiment=(average polarity={},average subjectivity={})'.format(avgpol,avgsub))

#tweet\_sub=[]

plt.hist(tweet\_sub, bins=20) #,normed=1,alpha=0.75)

plt.xlabel('subjectivity score')

plt.ylabel('tweets count')

plt.grid(True)

plt.savefig('subjectivity1.pdf')

plt.show()

def remove\_punc(tweet\_list):

new\_tweet\_list =[]

p = string.punctuation

table\_p = string.maketrans(p, len(p) \* " ")

for tweet in tweet\_list:

new\_tweet\_list.append(tweet.translate(table\_p))

return new\_tweet\_list

def remove\_digits(tweet\_list):

new\_tweet\_list =[]

d = string.digits

table\_d = string.maketrans(d, len(d) \* " ")

for tweet in tweet\_list:

new\_tweet\_list.append(tweet.translate(table\_d))

return new\_tweet\_list

#tweets2 =remove\_punc(remove\_digits(tweet\_text))

#print type(words), words[0:10], len(words)

from nltk.stem.lancaster import LancasterStemmer

ls = LancasterStemmer()

from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

from nltk.stem.snowball import SnowballStemmer

ss = SnowballStemmer("english")

outfile=open('StemmingList','w')

#for tweet in Tweets1.split():

# if word in tweet['text']:

# Tweets1[word.lower()]

#stemm=[]

outfile.write('PORTER LEMMATIZATION\n')

for tweet in Tweets1:

for word in tweet.split():

#print('Word: {}\tPorter:{} '.format(word.encode('utf-8').lower(), ps.stem(word).encode('utf-8')))

# print('Word: {}\tlancaster:{} '.format(word.encode('utf-8').lower(), ls.stem(word).encode('utf-8')))

#print('Word: {}\tSnowball:{} '.format(word.encode('utf-8').lower(), ss.stem(word).encode('utf-8')))

outfile.write(word.replace(word,ps.stem(word)).encode('utf-8')+'\t')

outfile.write('\n')

outfile.write('\n')

outfile.write('LANCASTER LEMMATIZATION\n')

for tweet in Tweets1:

for word in tweet.split():

# print('Word: {}\tlancaster:{} '.format(word.encode('utf-8').lower(), ls.stem(word).encode('utf-8')))

outfile.write(word.replace(word,ps.stem(word)).encode('utf-8')+'\t')

outfile.write('\n')

outfile.write('\n')

outfile.write('SNOWBALL LEMMATIZATION\n')

for tweet in Tweets1:

for word in tweet.split():

# print('Word: {}\tSnowball:{} '.format(word.encode('utf-8').lower(), ss.stem(word).encode('utf-8')))

# print('Word: {}\tlancaster:{} '.format(word.encode('utf-8').lower(), ls.stem(word).encode('utf-8')))

outfile.write(word.replace(word,ps.stem(word)).encode('utf-8')+'\t')

outfile.write('\n')

#print StemmingList

#print Tweets1

############################################################################

Trial ………………………………………………………….

# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import division, print\_function

import json

#import sys

from textblob import TextBlob

import matplotlib.pyplot as plt

import nltk

import string

#from wordcloud import WordCloud

tweets = []

tweet\_text = []

with open('tweet\_stream\_trump\_01.JSON', 'r') as f:

tweets = json.load(f)

print(len(tweets))

for tweet in tweets:

tweet\_text.append(tweet['text'].encode('ascii', 'replace').replace('?',' '))

def show\_sentiment(tweet\_list):

tweets\_sentiment = []

tweet\_sub = []

for tweet in tweet\_list:

tblob\_tweet = TextBlob(tweet)

tweets\_sentiment.append((tblob\_tweet.polarity, tblob\_tweet.subjectivity))

tweet\_sub.append(tblob\_tweet.subjectivity)

for y in tweets\_sentiment[0]:

print(type(y))

total\_polarity = 0

total\_subjectivity = 0

for x in tweets\_sentiment:

total\_polarity += x[0]

total\_subjectivity += x[1]

print ('Average polarity = {}'.format(total\_polarity/len(tweets\_sentiment)))

print ('Average subjectivity = {}'.format(total\_subjectivity/len(tweets\_sentiment)))

plt.hist(tweet\_sub, bins=20)

plt.xlabel('subjectivity score')

plt.ylabel('tweet count')

plt.grid(True)

def remove\_punc(tweet\_list):

new\_tweet\_list =[]

p = string.punctuation

table\_p = string.maketrans(p, len(p) \* " ")

for tweet in tweet\_list:

new\_tweet\_list.append(tweet.translate(table\_p))

return new\_tweet\_list

def remove\_digits(tweet\_list):

new\_tweet\_list =[]

d = string.digits

table\_d = string.maketrans(d, len(d) \* " ")

for tweet in tweet\_list:

new\_tweet\_list.append(tweet.translate(table\_d))

return new\_tweet\_list

tweets2 =remove\_punc(remove\_digits(tweet\_text))

show\_sentiment(tweets2)

from nltk.stem.lancaster import LancasterStemmer

ls = LancasterStemmer()

from nltk.stem.porter import PorterStemmer

ps = PorterStemmer()

from nltk.stem.snowball import SnowballStemmer

ss = SnowballStemmer("english")

from nltk.stem import WordNetLemmatizer

wnL=WordNetLemmatizer()

all\_tweet\_words\_raw = ''

all\_tweet\_words\_ls = ''

all\_tweet\_words\_ps = ''

all\_tweet\_words\_ss = ''

all\_tweet\_words\_wnl = ''

for tweet in tweets2:

for word in tweet.split():

all\_tweet\_words\_raw +=' {}'.format(word.lower())

all\_tweet\_words\_ls +=' {}'.format(ls.stem(word).lower())

all\_tweet\_words\_ps +=' {}'.format(ps.stem(word).lower())

all\_tweet\_words\_ss +=' {}'.format(ss.stem(word).lower())

all\_tweet\_words\_wnl +=' {}'.format(wnL.lemmatize(word).lower())

print("length of all tweet words={}".format(len(all\_tweet\_words\_raw)))

words2 = []

def remove\_stopwords(word\_list):

words = nltk.word\_tokenize(word\_list)

print("Tokenized words={}".format(words))

freq = nltk.FreqDist(words)

freq.plot(25)

stopwords = nltk.corpus.stopwords.words('english')

more\_stopwords = stopwords +['trump','donald','president','amp']

print(more\_stopwords)

for w in words:

if w not in more\_stopwords and len(w) > 2:

words2.append(w)

freq2 = nltk.FreqDist(words2)

print('raw text')

freq.plot(30)

print('text without stopwords')

freq2.plot(30)

return words2

"""

text1= ''

for word in remove\_stopwords(all\_tweet\_words\_wnl):

text1 += ' {}'.format(word)

wordcloud = WordCloud(max\_font\_size=40).generate(text1)

plt.figure()

plt.imshow(wordcloud)

plt.axis("off")

plt.show()

image = wordcloud.to\_image()

image.show()

#NMF

vectorizer = TfidfVectorizer(stop\_words='english', min\_df=2)

doc\_term\_matrix = vectorizer.fit\_transform(words2)

print(doc\_term\_matrix.shape)

vocab = vectorizer.get\_feature\_names() # list of unique vocab, we will use this later

print(len(vocab)), '# of unique words'

print(vocab[-10:])

print(vocab[:10])

#print dtm.shape

print('num of documents, num of unique words')

print(doc\_term\_matrix.shape)

num\_topics =5

clf = decomposition.NMF(n\_components=num\_topics, random\_state=1)

doctopic = clf.fit\_transform(doc\_term\_matrix)

print(num\_topics, clf.reconstruction\_err\_)

topic\_words = []

num\_top\_words = 5

print(vocab[100])

for topic in clf.components\_:

word\_idx = np.argsort(topic)[::-1][:num\_top\_words]

print(word\_idx)

for idx in word\_idx:

print(vocab[idx])

print(vocab[idx])

print('\_\_++++\_\_' \* 10)

for t in range(len(topic\_words)):

print("Topic {}: {}".format(t, ' '.join(topic\_words[t][:15])))

from sklearn import decomposition

for n in range(1, 10):

num\_topics = 5\*n

num\_top\_words = 5

clf = decomposition.NMF(n\_components=num\_topics, random\_state=1)

doctopic = clf.fit\_transform(doc\_term\_matrix)

print(num\_topics, clf.reconstruction\_err\_)

#LDA

logging.basicConfig(format='%(levelname)s : %(message)s', level=logging.INFO)

logging.root.level = logging.INFO

# ipython sometimes messes up the logging setup; restore

docs = []

corpus=words2

for word in corpus:

docs.append(word.strip().split())

#for corp in corpus:

# print(len(corp), corp[:10])

dic = corpora.Dictionary(docs)

print(dic)

corpus1 = [dic.doc2bow(text) for text in docs]

print(type(corpus1), len(corpus1))

tfidf = models.TfidfModel(corpus1)

print(type(tfidf))

tfidfcorpus=tfidf[corpus1]

print(type(tfidfcorpus))

num\_topics = 5

model = models.ldamodel.LdaModel(tfidfcorpus,

id2word=dic,

num\_topics=num\_topics)

print("LDA model")

topics\_found = model.print\_topics(7)

counter = 1

for t in topics\_found:

print("Topic #{} {}".format(counter, t))

counter += 1

model.print\_topics()"""