import numpy as np

import pandas as pd

from sklearn import preprocessing

from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer

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

from nltk.corpus import stopwords

from nltk.stem.snowball import SnowballStemmer

import matplotlib

from matplotlib import pyplot as plt

# import seaborn as sns

%matplotlib inline

%config InlineBackend.figure\_format = 'retina'

data = pd.read\_csv('RAW\_interactions.csv')

def remove\_punctuation(text):

'''a function for removing punctuation'''

import string

# replacing the punctuations with no space,

# which in effect deletes the punctuation marks

translator = str.maketrans('', '', string.punctuation)

# return the text stripped of punctuation marks

return text.translate(translator)

data.dropna(how = 'all', inplace = False)

data.head(5)

x1 = data['review'] = data['review'].astype(str).apply(remove\_punctuation)

data.head(10)

import nltk

nltk.download()

# extracting the stopwords from nltk library

sw = stopwords.words('english')

# displaying the stopwords

np.array(sw)

def stopwords(text):

'''a function for removing the stopword'''

# removing the stop words and lowercasing the selected words

text = [word.lower() for word in text.split() if word.lower() not in sw]

# joining the list of words with space separator

return " ".join(text)

data['review'] = data['review'].apply(stopwords)

data.head(10)

# create a count vectorizer object

count\_vectorizer = CountVectorizer()

# fit the count vectorizer using the text data

count\_vectorizer.fit(data['review'])

# collect the vocabulary items used in the vectorizer

dictionary = count\_vectorizer.vocabulary\_.items()

# lists to store the vocab and counts

vocab = []

count = []

# iterate through each vocab and count append the value to designated lists

for key, value in dictionary:

vocab.append(key)

count.append(value)

# store the count in panadas dataframe with vocab as index

vocab\_bef\_stem = pd.Series(count, index=vocab)

# sort the dataframe

#vocab\_bef\_stem = vocab\_bef\_stem.sort\_values(ascending=False)

top\_vacab = vocab\_bef\_stem.head(20)

top\_vacab.plot(kind = 'barh', figsize=(5,10))

# create an object of stemming function

stemmer = SnowballStemmer("english")

def stemming(text):

'''a function which stems each word in the given text'''

text = [stemmer.stem(word) for word in text.split()]

return " ".join(text)

data['review'] = data['review'].apply(stemming)

data.head(10)

‘’’’

New approach at NLP

‘’’’

import re

from nltk.corpus import stopwords

# for Stemming propose

from nltk.stem.porter import PorterStemmer

# Initialize empty array

# to append clean text

corpus = []

# 1000 (reviews) rows to clean

for i in range(0, 1000):

# column : "Review", row ith

review = re.sub('[^a-zA-Z]', ' ', data['review'][i])

# convert all cases to lower cases

review = review.lower()

# split to array(default delimiter is " ")

review = review.split()

# creating PorterStemmer object to

# take main stem of each word

ps = PorterStemmer()

# loop for stemming each word

# in string array at ith row

review = [ps.stem(word) for word in review

if not word in set(stopwords.words('english'))]

# rejoin all string array elements

# to create back into a string

review = ' '.join(review)

# append each string to create

# array of clean text

corpus.append(review)

# Creating the Bag of Words model

from sklearn.feature\_extraction.text import CountVectorizer

# To extract max 1500 feature.

# "max\_features" is attribute to

# experiment with to get better results

cv = CountVectorizer(max\_features = 1500)

# X contains corpus (dependent variable)

X = cv.fit\_transform(corpus).toarray()

# y contains answers if review

# is positive or negative

y = data.iloc[:, 1].values

dictionary = cv.vocabulary\_.items()

vocab = []

count = []

# iterate through each vocab and count append the value to designated lists

for key, value in dictionary:

vocab.append(key)

count.append(value)

# store the count in panadas dataframe with vocab as index

vocab\_bef\_stem = pd.Series(count, index=vocab)

# sort the dataframe

#vocab\_bef\_stem = vocab\_bef\_stem.sort\_values(ascending=False)

top\_vacab = vocab\_bef\_stem.to\_frame()

top\_vacab.plot(kind = 'barh', figsize=(5,10))