|  |  |  |
| --- | --- | --- |
| Vocabulary size | Alpha (laplace smoothing) | Accuracy on 20% validation set |
| 3000 | 0.00001 | **83.98** |
| 3000 | 5 | **84.02** |
| 5000 | 0.00001 | **84.16** |
| 5000 | 5 | **84.20** |

**Code:  
1:**

import pandas as pd

import re

import nltk

import numpy as np

from bs4 import BeautifulSoup

from nltk.corpus import stopwords

# Run the BeautifulSoup object on a single movie review

pd.options.mode.chained\_assignment = None

train = pd.read\_csv('labeledTrainData.tsv', header=0, delimiter='\t', quoting=3)

train.head()

reviews\_list=[]

stops = set(stopwords.words('english'))

def review\_to\_words(raw\_review):

xyz = BeautifulSoup(raw\_review,'html.parser').get\_text()

xyz = re.sub('[^a-zA-Z]',' ',xyz)

xyz = xyz.lower().split()

stops = set(stopwords.words('english'))

xyz = [w for w in xyz if not w in stops]

return " ".join(xyz)

for x in range(0,25000):

raw\_review = train["review"][x]

reviews\_list.append(review\_to\_words(raw\_review))

**2:**

X=[]

reviews\_test = []

for x in range(0,20000):

X.append(reviews\_list[x])

for x in range(20000,25000):

reviews\_test.append(reviews\_list[x])

print (len(X))

print (len(reviews\_test))

**3:**

# Initialize the "CountVectorizer" object, which is scikit-learn's

# bag of words tool.

from sklearn.feature\_extraction.text import CountVectorizer

vectorizer = CountVectorizer(analyzer = "word",

tokenizer = None,

preprocessor = None,

stop\_words = None,

max\_features = 3000)

X = vectorizer.fit\_transform(X)

X = X.toarray()

print(X.shape)

**4:**

sentiment\_training = []

sentiment\_test = []

for x in range (0,20000):

sentiment\_training.append(train["sentiment"][x])

for x in range(20000,25000):

sentiment\_test.append(train["sentiment"][x])

**5:**

print(len(sentiment\_training))

print(len(X))

**6:**

from sklearn.naive\_bayes import MultinomialNB

import numpy as np

clf = MultinomialNB(alpha=5.0) #alpha is for smoothing so and we variate it to see its effects

clf.fit(X, np.array(sentiment\_training))

**7:**

test\_vectorized = vectorizer.transform(reviews\_test).toarray()

print(clf.predict(test\_vectorized))

prediction = clf.predict(test\_vectorized)

count =0

for x in range (0,5000):

if prediction[x]==sentiment\_test[x]:

count= count +1

print(count/50)