

Assignment2_Amazon_Fine_Food_Reviews

October 19, 2018

1 Amazon Find Food Reviews Analysis

- The Amazon Fine Food Reviews dataset consists of reviews of fine foods from Amazon.

Number of reviews: 568,454 Number of users: 256,059 Number of products: 74,258 Timespan: Oct 1999 - Oct 2012 Number of Attributes/Columns in data: 10

Attribute Information:

1. Id
2. ProductId - unique identifier for the product
3. UserId - unique identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful ---> Yes
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not ---> Yes/NO
7. Score - rating between 1 and 5
8. Time - timestamp for the review
9. Summary - brief summary of the review
10. Text - text of the review

Objective:

Given a review, determine whether the review is positive (Rating of 4 or 5) or negative (rating of 1 or 2).

Que:- How to determine if a review is positive or negative?

Ans:-

1. A rating of 4 or 5 could be considered a positive review.
2. A review of 1 or 2 could be considered negative.
3. A review of 3 is neutral and ignored.

```
In [2]: import warnings
        warnings.filterwarnings('ignore')
```

```
In [3]: %matplotlib inline

import sqlite3
import pandas as pd
import numpy as np
```

```

import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
import nltk
from nltk.stem.porter import PorterStemmer

con = sqlite3.connect('./amazon-fine-food-reviews/database.sqlite')

filtered_data = pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3
""", con
)

def partition(x):
    if x<3:
        return "negative"
    return "positive"

actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative

```

```

In [4]: print(filtered_data.shape)
        filtered_data.head()

```

(525814, 10)

```

Out[4]:

```

	Id	ProductId	UserId	ProfileName	\
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
2	3	B000LQOCHO	ABXLMWJIXXAIN	Natalia Corres	"Natalia Corres"
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham	"M. Wassir"

	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	\
0	1	1	positive	1303862400	
1	0	0	negative	1346976000	
2	1	1	positive	1219017600	

3	3	3	negative	1307923200
4	0	0	positive	1350777600

	Summary	Text
0	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	"Delight" says it all	This is a confection that has been around a fe...
3	Cough Medicine	If you are looking for the secret ingredient i...
4	Great taffy	Great taffy at a great price. There was a wid...

2 Data Cleaning: Deduplication

```
In [5]: display = pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
Order By ProductID
""", con)
display.head()
```

```
Out[5]:
```

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator \
0	78445	B000HDL1RQ	AR5J8UI46CURR	Geetha Krishnan	2
1	138317	B000HDOPYC	AR5J8UI46CURR	Geetha Krishnan	2
2	138277	B000HDOPYM	AR5J8UI46CURR	Geetha Krishnan	2
3	73791	B000HDOPZG	AR5J8UI46CURR	Geetha Krishnan	2
4	155049	B000PAQ75C	AR5J8UI46CURR	Geetha Krishnan	2

	HelpfulnessDenominator	Score	Time \
0	2	5	1199577600
1	2	5	1199577600
2	2	5	1199577600
3	2	5	1199577600
4	2	5	1199577600

	Summary \
0	LOACKER QUADRATINI VANILLA WAFERS
1	LOACKER QUADRATINI VANILLA WAFERS
2	LOACKER QUADRATINI VANILLA WAFERS
3	LOACKER QUADRATINI VANILLA WAFERS
4	LOACKER QUADRATINI VANILLA WAFERS

	Text
0	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
1	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
2	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
3	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
4	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...

```
In [6]: sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=False)
```

```
In [7]: final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first')
final.shape
```

```
Out[7]: (364173, 10)
```

```
In [8]: (final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

```
Out[8]: 69.25890143662969
```

Observation:- It was also seen that in two rows given below the value of HelpfulnessNumerator is greater than HelpfulnessDenominator which is not practically possible hence these two rows too are removed from calculations

```
In [9]: display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)
```

```
display.head()
```

```
Out[9]:
```

	Id	ProductId	UserId	ProfileName	\
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens	"Jeanne"
1	44737	B001EQ55RW	A2V0I904FH7ABY		Ram

	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	\
0	3	1	5	1224892800	
1	3	2	4	1212883200	

	Summary	\
0	Bought This for My Son at College	
1	Pure cocoa taste with crunchy almonds inside	

	Text
0	My son loves spaghetti so I didn't hesitate or...
1	It was almost a 'love at first bite' - the per...

```
In [10]: final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

```
In [11]: print(final.shape)
```

```
final['Score'].value_counts()
```

```
(364171, 10)
```

```
Out[11]: positive    307061
negative    57110
Name: Score, dtype: int64
```

3 Bag of Words (BOW)

```
In [12]: positive_data = final[final["Score"] == "positive"].sample(n = 2000)
        negative_data = final[final["Score"] == "negative"].sample(n = 2000)
        final_4000 = pd.concat([positive_data, negative_data])
```

```
In [13]: score_4000 = final_4000['Score']
```

```
In [14]: score_4000.shape
```

```
Out[14]: (4000,)
```

```
In [15]: final_4000.shape
```

```
Out[15]: (4000, 10)
```

```
In [16]: count_vect = CountVectorizer()
        final_counts = count_vect.fit_transform(final_4000['Text'].values)
```

```
In [17]: type(final_counts)
```

```
Out[17]: scipy.sparse.csr.csr_matrix
```

```
In [18]: final_counts.get_shape()
```

```
Out[18]: (4000, 13853)
```

4 Bi-grams and n-grams

```
In [19]: count_vect = CountVectorizer(ngram_range=(1,2) )
        final_bigram_counts = count_vect.fit_transform(final_4000['Text'].values)
```

```
In [20]: final_bigram_counts.get_shape()
```

```
Out[20]: (4000, 144944)
```

```
In [21]: from sklearn.preprocessing import StandardScaler
```

```
        standard_data = StandardScaler(with_mean = False).fit_transform(final_bigram_counts)
        standard_data.shape
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:
  warnings.warn(msg, DataConversionWarning)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\validation.py:475: DataConversionWarning:
  warnings.warn(msg, DataConversionWarning)
```

```
Out[21]: (4000, 144944)
```

```
In [22]: type(standard_data)
```

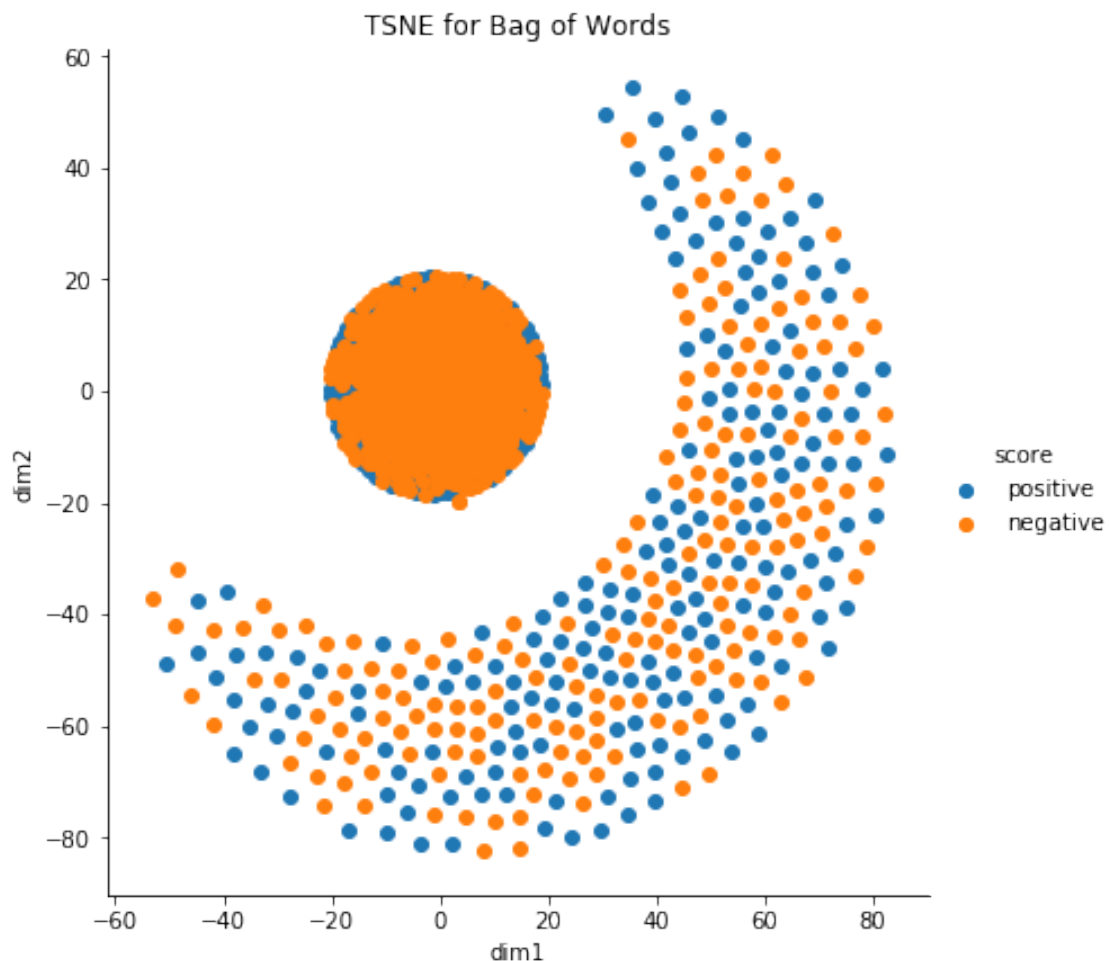
```
Out[22]: scipy.sparse.csr.csr_matrix
```

```
In [23]: standard_data = standard_data.todense()  
         # convert sparse to dense as tsne takes dense vector
```

```
In [24]: type(standard_data)
```

```
Out[24]: numpy.matrixlib.defmatrix.matrix
```

```
In [24]: from sklearn.manifold import TSNE  
         model = TSNE(n_components=2, random_state=0, perplexity = 30)  
  
         tsne_data = model.fit_transform(standard_data)  
  
         tsne_data = np.vstack((tsne_data.T, score_4000)).T  
         tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "score"))  
  
         sns.FacetGrid(tsne_df, hue="score", size=6).map(plt.scatter, 'dim1', 'dim2').add_legend()  
         plt.title("TSNE for Bag of Words")  
         plt.show()
```



Obeservation:-

Here, we are unable to classify the +ve and -ve points because it overlaps each other.

5 TF-IDF

```
In [25]: tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
        final_tf_idf = tf_idf_vect.fit_transform(final_4000['Text'].values)
```

```
In [26]: from sklearn.preprocessing import StandardScaler
        std = StandardScaler(with_mean = False)
        standard_data = std.fit_transform(final_tf_idf)
```

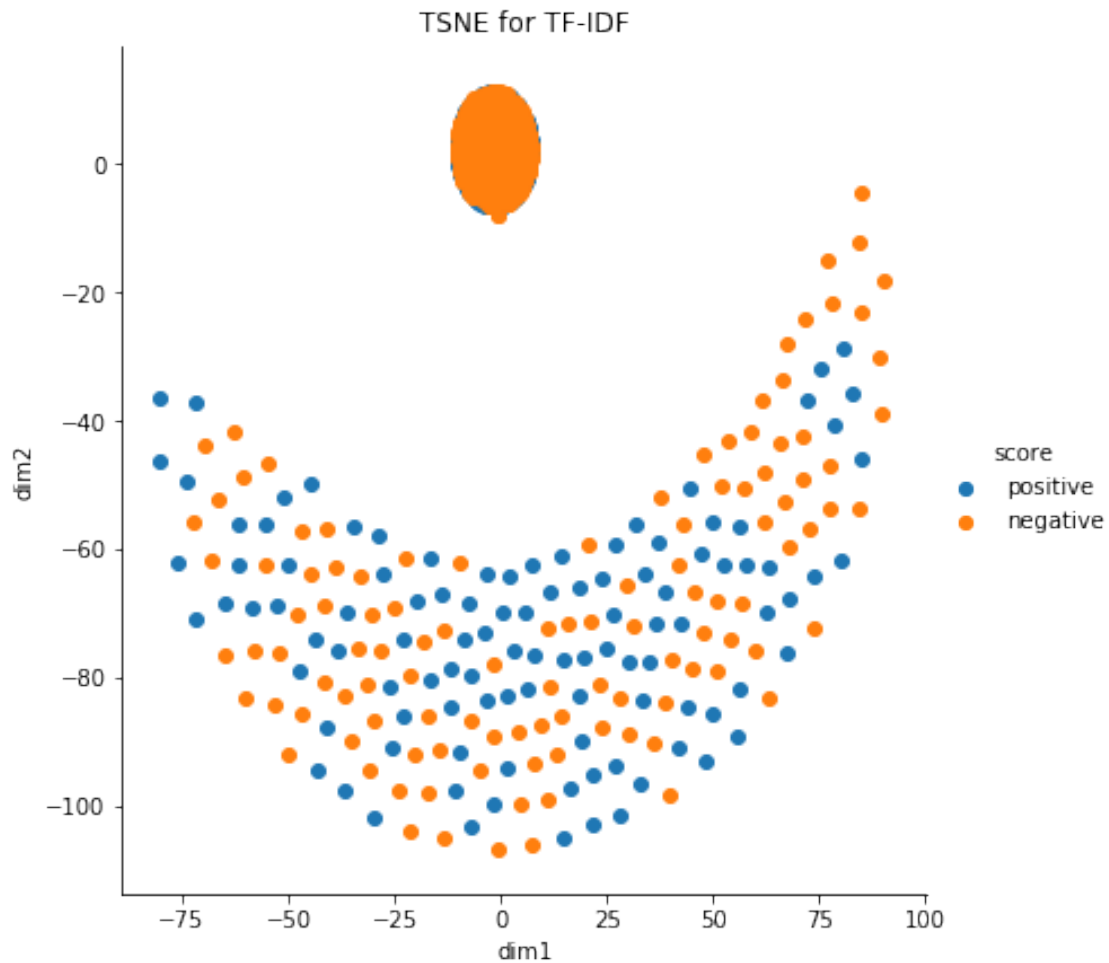
```
In [27]: standard_data = standard_data.todense()
```

```
In [28]: standard_data.shape
```

```
Out[28]: (4000, 144944)
```

```
In [29]: from sklearn.manifold import TSNE
        model = TSNE(n_components = 2, perplexity = 50)
        tsne_data = model.fit_transform(standard_data)

        tsne_data = np.vstack((tsne_data.T, score_4000)).T
        tsne_df = pd.DataFrame(data = tsne_data, columns = ("dim1", "dim2", "score"))
        sns.FacetGrid(tsne_df, hue = "score", size = 6).map(plt.scatter, "dim1", "dim2").add_
        plt.title("TSNE for TF-IDF")
        plt.show()
```



Observation:-

As we saw earlier, it overlaps +ve and -ve points same as Bag of Words.

6 Word2Vec

```
In [29]: import gensim
          from gensim.models import Word2Vec
          from gensim.models import KeyedVectors
          import pickle

          model = KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', bi
```

C:\ProgramData\Anaconda3\lib\site-packages\gensim\utils.py:1197: UserWarning: detected Windows warnings.warn("detected Windows; aliasing chunkize to chunkize_serial")

```
In [30]: model.wv['computer']
```



```
Out[30]: array([ 1.07421875e-01, -2.01171875e-01,  1.23046875e-01,  2.11914062e-01,
-9.13085938e-02,  2.16796875e-01, -1.31835938e-01,  8.30078125e-02,
 2.02148438e-01,  4.78515625e-02,  3.66210938e-02, -2.45361328e-02,
 2.39257812e-02, -1.60156250e-01, -2.61230469e-02,  9.71679688e-02,
-6.34765625e-02,  1.84570312e-01,  1.70898438e-01, -1.63085938e-01,
-1.09375000e-01,  1.49414062e-01, -4.65393066e-04,  9.61914062e-02,
 1.68945312e-01,  2.60925293e-03,  8.93554688e-02,  6.49414062e-02,
 3.56445312e-02, -6.93359375e-02, -1.46484375e-01, -1.21093750e-01,
-2.27539062e-01,  2.45361328e-02, -1.24511719e-01, -3.18359375e-01,
-2.20703125e-01,  1.30859375e-01,  3.66210938e-02, -3.63769531e-02,
-1.13281250e-01,  1.95312500e-01,  9.76562500e-02,  1.26953125e-01,
 6.59179688e-02,  6.93359375e-02,  1.02539062e-02,  1.75781250e-01,
-1.68945312e-01,  1.21307373e-03, -2.98828125e-01, -1.15234375e-01,
 5.66406250e-02, -1.77734375e-01, -2.08984375e-01,  1.76757812e-01,
 2.38037109e-02, -2.57812500e-01, -4.46777344e-02,  1.88476562e-01,
 5.51757812e-02,  5.02929688e-02, -1.06933594e-01,  1.89453125e-01,
-1.16210938e-01,  8.49609375e-02, -1.71875000e-01,  2.45117188e-01,
-1.73828125e-01, -8.30078125e-03,  4.56542969e-02, -1.61132812e-02,
 1.86523438e-01, -6.05468750e-02, -4.17480469e-02,  1.82617188e-01,
 2.20703125e-01, -1.22558594e-01, -2.55126953e-02, -3.08593750e-01,
 9.13085938e-02,  1.60156250e-01,  1.70898438e-01,  1.19628906e-01,
 7.08007812e-02, -2.64892578e-02, -3.08837891e-02,  4.06250000e-01,
-1.01562500e-01,  5.71289062e-02, -7.26318359e-03, -9.17968750e-02,
-1.50390625e-01, -2.55859375e-01,  2.16796875e-01, -3.63769531e-02,
 2.24609375e-01,  8.00781250e-02,  1.56250000e-01,  5.27343750e-02,
 1.50390625e-01, -1.14746094e-01, -8.64257812e-02,  1.19140625e-01,
-7.17773438e-02,  2.73437500e-01, -1.64062500e-01,  7.29370117e-03,
 4.21875000e-01, -1.12792969e-01, -1.35742188e-01, -1.31835938e-01,
-1.37695312e-01, -7.66601562e-02,  6.25000000e-02,  4.98046875e-02,
-1.91406250e-01, -6.03027344e-02,  2.27539062e-01,  5.88378906e-02,
-3.24218750e-01,  5.41992188e-02, -1.35742188e-01,  8.17871094e-03,
-5.24902344e-02, -1.74713135e-03, -9.81445312e-02, -2.86865234e-02,
 3.61328125e-02,  2.15820312e-01,  5.98144531e-02, -3.08593750e-01,
-2.27539062e-01,  2.61718750e-01,  9.86328125e-02, -5.07812500e-02,
 1.78222656e-02,  1.31835938e-01, -5.35156250e-01, -1.81640625e-01,
 1.38671875e-01, -3.10546875e-01, -9.71679688e-02,  1.31835938e-01,
-1.16210938e-01,  7.03125000e-02,  2.85156250e-01,  3.51562500e-02,
-1.01562500e-01, -3.75976562e-02,  1.41601562e-01,  1.42578125e-01,
-5.68847656e-02,  2.65625000e-01, -2.09960938e-01,  9.64355469e-03,
-6.68945312e-02, -4.83398438e-02, -6.10351562e-02,  2.45117188e-01,
-9.66796875e-02,  1.78222656e-02, -1.27929688e-01, -4.78515625e-02,
-7.26318359e-03,  1.79687500e-01,  2.78320312e-02, -2.10937500e-01,
-1.43554688e-01, -1.27929688e-01,  1.73339844e-02, -3.60107422e-03,
-2.04101562e-01,  3.63159180e-03, -1.19628906e-01, -6.15234375e-02,
 5.93261719e-02, -3.23486328e-03, -1.70898438e-01, -3.14941406e-02,
-8.88671875e-02, -2.89062500e-01,  3.44238281e-02, -1.87500000e-01,
 2.94921875e-01,  1.58203125e-01, -1.19628906e-01,  7.61718750e-02,
 6.39648438e-02, -4.68750000e-02, -6.83593750e-02,  1.21459961e-02,
```

```

-1.44531250e-01, 4.54101562e-02, 3.68652344e-02, 3.88671875e-01,
1.45507812e-01, -2.55859375e-01, -4.46777344e-02, -1.33789062e-01,
-1.38671875e-01, 6.59179688e-02, 1.37695312e-01, 1.14746094e-01,
2.03125000e-01, -4.78515625e-02, 1.80664062e-02, -8.54492188e-02,
-2.48046875e-01, -3.39843750e-01, -2.83203125e-02, 1.05468750e-01,
-2.14843750e-01, -8.74023438e-02, 7.12890625e-02, 1.87500000e-01,
-1.12304688e-01, 2.73437500e-01, -3.26171875e-01, -1.77734375e-01,
-4.24804688e-02, -2.69531250e-01, 6.64062500e-02, -6.88476562e-02,
-1.99218750e-01, -7.03125000e-02, -2.43164062e-01, -3.66210938e-02,
-7.37304688e-02, -1.77734375e-01, 9.17968750e-02, -1.25000000e-01,
-1.65039062e-01, -3.57421875e-01, -2.85156250e-01, -1.66992188e-01,
1.97265625e-01, -1.53320312e-01, 2.31933594e-02, 2.06054688e-01,
1.80664062e-01, -2.74658203e-02, -1.92382812e-01, -9.61914062e-02,
-1.06811523e-02, -4.73632812e-02, 6.54296875e-02, -1.25732422e-02,
1.78222656e-02, -8.00781250e-02, -2.59765625e-01, 9.37500000e-02,
-7.81250000e-02, 4.68750000e-02, -2.22167969e-02, 1.86767578e-02,
3.11279297e-02, 1.04980469e-02, -1.69921875e-01, 2.58789062e-02,
-3.41796875e-02, -1.44042969e-02, -5.46875000e-02, -8.78906250e-02,
1.96838379e-03, 2.23632812e-01, -1.36718750e-01, 1.75781250e-01,
-1.63085938e-01, 1.87500000e-01, 3.44238281e-02, -5.63964844e-02,
-2.27689743e-05, 4.27246094e-02, 5.81054688e-02, -1.07910156e-01,
-3.88183594e-02, -2.69531250e-01, 3.34472656e-02, 9.81445312e-02,
5.63964844e-02, 2.23632812e-01, -5.49316406e-02, 1.46484375e-01,
5.93261719e-02, -2.19726562e-01, 6.39648438e-02, 1.66015625e-02,
4.56542969e-02, 3.26171875e-01, -3.80859375e-01, 1.70898438e-01,
5.66406250e-02, -1.04492188e-01, 1.38671875e-01, -1.57226562e-01,
3.23486328e-03, -4.80957031e-02, -2.48046875e-01, -6.20117188e-02],
dtype=float32)

```

```
In [31]: model.wv.similarity('woman', 'man')
```

```
Out[31]: 0.7664012230995352
```

```
In [32]: model.wv.similarity('tasty', 'tast')
```

```
Out[32]: 0.440350541900889
```

```
In [45]: i=0
         list_of_sent=[]
         for sent in final['Text'].values:
             list_of_sent.append(sent.split())
```

```
In [46]: print(final['Text'].values[0])
         print("*****")
         print(list_of_sent[0])
```

```
this witty little book makes my son laugh at loud. i recite it in the car as we're driving along
*****
['this', 'witty', 'little', 'book', 'makes', 'my', 'son', 'laugh', 'at', 'loud.', 'i', 'recite
```

```

In [47]: w2v_model=gensim.models.Word2Vec(list_of_sent,min_count=5,size=50, workers=4)

In [48]: w2v_words = list(w2v_model.wv.vocab)
          print("number of words that occurred minimum 5 times ",len(w2v_words))
          print("sample words ", w2v_words[0:50])

number of words that occurred minimum 5 times 94472
sample words ['this', 'witty', 'little', 'book', 'makes', 'my', 'son', 'laugh', 'at', 'loud.'

In [49]: w2v_model.wv.most_similar('tasty')

Out[49]: [('tasty,', 0.8943316340446472),
           ('delicious', 0.8842543363571167),
           ('satisfying', 0.8765422105789185),
           ('yummy', 0.8709976673126221),
           ('tastey', 0.8570417165756226),
           ('versatile', 0.810888409614563),
           ('hearty', 0.8050057888031006),
           ('flavorful', 0.803313136100769),
           ('filling,', 0.7926658391952515),
           ('nutritious', 0.7917579412460327)]

In [50]: w2v_model.wv.most_similar('like')

Out[50]: [('like,', 0.7373756766319275),
           ('prefer', 0.6679574251174927),
           ('enjoy', 0.6527254581451416),
           ('resemble', 0.6477119326591492),
           ('like.', 0.6435332298278809),
           ('like).', 0.6311202049255371),
           ('miss', 0.6239169836044312),
           ('love', 0.6085191965103149),
           ('dislike', 0.585246205329895),
           ('mean', 0.5844687223434448)]

```

7 Avg W2V

```

In [56]: sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
          for sent in list_of_sent: # for each review/sentence
              sent_vec = np.zeros(50) # as word vectors are of zero length
              cnt_words = 0; # num of words with a valid vector in the sentence/review
              for word in sent: # for each word in a review/sentence
                  try:
                      vec = w2v_model.wv[word]
                      sent_vec += vec
                      cnt_words += 1
                  except:

```

```

        pass
    sent_vec /= cnt_words
    sent_vectors.append(sent_vec)
print(len(sent_vectors))
print(len(sent_vectors[0]))

```

364171
50

```
In [57]: sent_vectors = sent_vectors[0:4000]
        len(sent_vectors)
```

Out[57]: 4000

```
In [58]: from sklearn.manifold import TSNE
        model = TSNE(n_components=2, random_state=15)

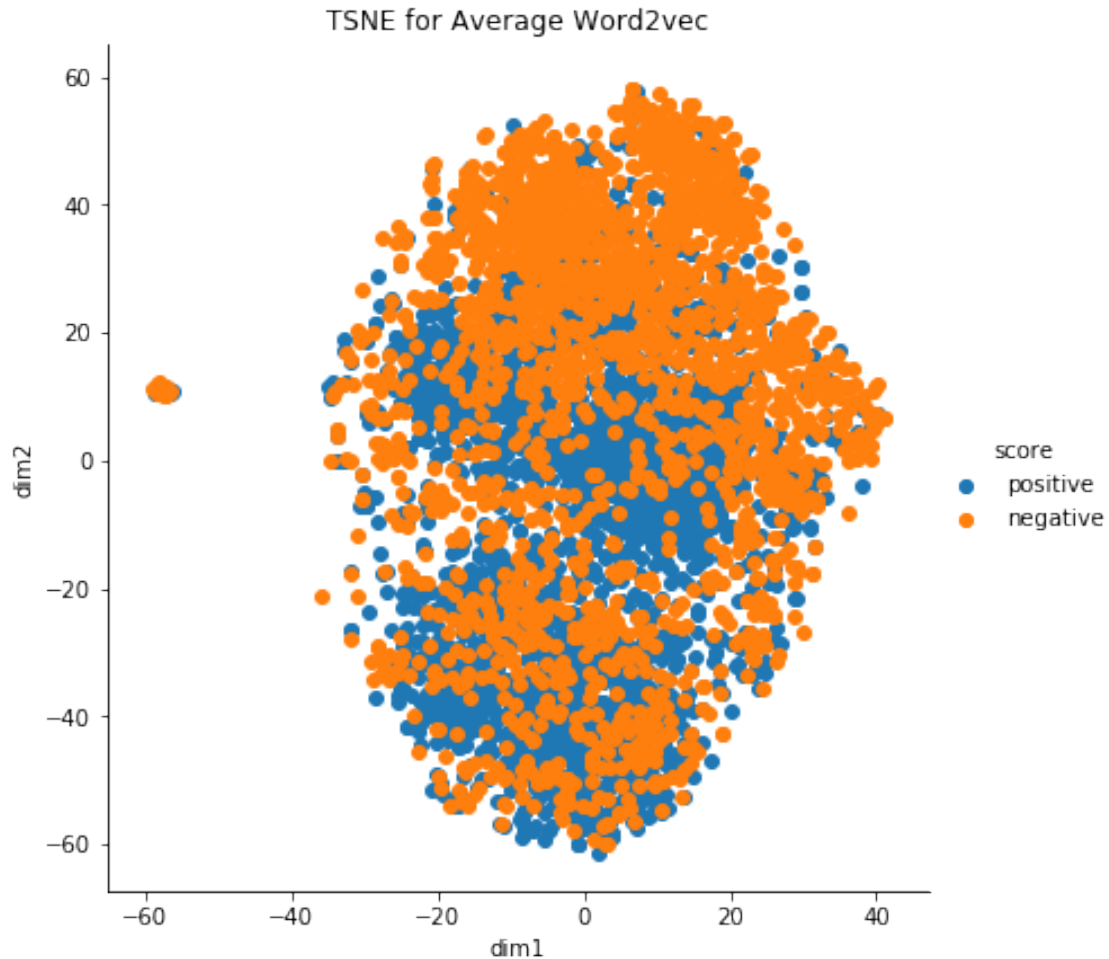
        tsne_data = model.fit_transform(sent_vectors)

        tsne_data = np.vstack((tsne_data.T, score_4000)).T
        tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "score"))

        # Plotting the result of tsne
        sns.FacetGrid(tsne_df, hue="score", size=6).map(plt.scatter, 'dim1', 'dim2').add_legend()
        plt.title("TSNE for Average Word2vec")
        plt.show()

```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` parameter is deprecated. Use the `height` and `width` parameters instead.
warnings.warn(msg, UserWarning)



Observation:-

We can't classify from the Avg W2V because of all +ve and -ve reviews are not well separated.

8 TFIDF Word2Vec

```
In [57]: model = TfidfVectorizer()
         tf_idf_matrix = model.fit_transform(final['Text'].values)
         # we are converting a dictionary with word as a key, and the idf as a value
         dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))

In [58]: # TF-IDF weighted Word2Vec
         tfidf_feat = tf_idf_vect.get_feature_names()

         tfidf_sent_vectors = []; # the tfidf-w2v for each sentence/review is stored in this l
         row=0;
         for sent in list_of_sent: # for each review/sentence
```

```

sent_vec = np.zeros(50) # as word vectors are of zero length
weight_sum = 0; # num of words with a valid vector in the sentence/review
for word in sent: # for each word in a review/sentence
    try:
        vec = w2v_model.wv[word]
        # obtain the tf_idfidf of a word in a sentence/review
        # tf_idf = final_tf_idf[row, tfidf_feat.index(word)]
        tf_idf = dictionary[word]*sent.count(word)
        sent_vec += (vec * tf_idf)
        weight_sum += tf_idf
    except:
        pass
sent_vec /= weight_sum
tfidf_sent_vectors.append(sent_vec)
row += 1

```

```
In [59]: len(tfidf_sent_vectors)
```

```
Out[59]: 364171
```

```
In [60]: tfidf_sent_vectors = tfidf_sent_vectors[0:4000]
len(tfidf_sent_vectors)
```

```
Out[60]: 4000
```

```
In [61]: tfidf_sent_vectors = np.nan_to_num(tfidf_sent_vectors)
```

```
In [62]: from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=15, perplexity=20, n_iter=2000)
```

```
tsne_data = model.fit_transform(tfidf_sent_vectors)
```

```
tsne_data = np.vstack((tsne_data.T, score_4000)).T
```

```
tsne_df = pd.DataFrame(data=tsne_data, columns=("dim1", "dim2", "score"))
```

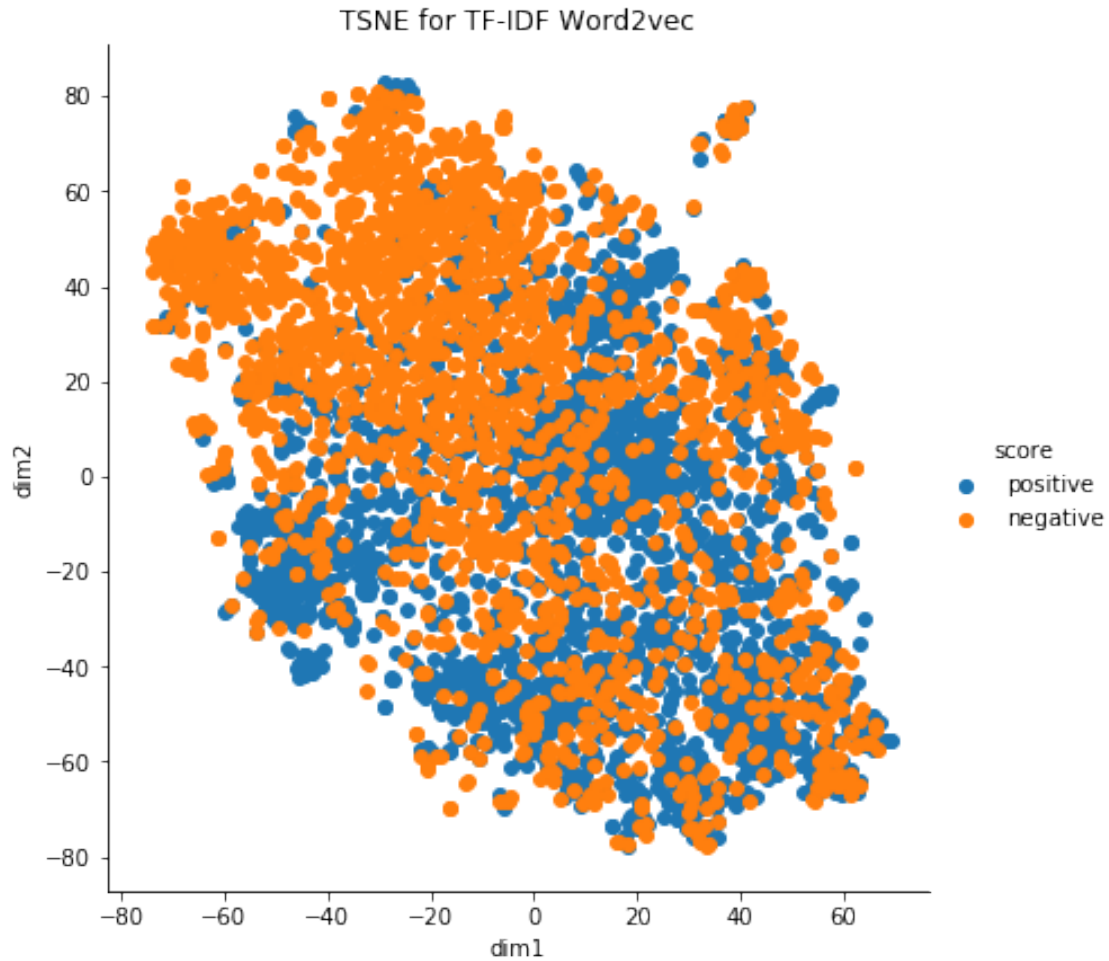
```
# Plotting the result of tsne
```

```
sns.FacetGrid(tsne_df, hue="score", size=6).map(plt.scatter, 'dim1', 'dim2').add_legend
```

```
plt.title("TSNE for TF-IDF Word2vec")
```

```
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:230: UserWarning: The `size` parameter
warnings.warn(msg, UserWarning)
```



Observation:-

The +ve and -ve reviews are overlapped each other, it looks same as bow, tfidf and avg word2vec.

Conclusion:-

1. As we saw TSNE representation, all the +ve and -ve reviews are overlapped each other.
2. We can not simply draw a plane to separate -ve and +ve reviews.