Assignment2

October 4, 2018

In [51]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
import pickle,pprint

import sqlite3
import pandas as pd
import numpy as np
import nltk


import re


from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
import matplotlib.pyplot as plt




from sklearn.feature_extraction.text import TfidfVectorizer


from gensim.models import Word2Vec
```

In [10]:
```
con=sqlite3.connect('database.sqlite')
```

In [11]:
```
raw_data=pd.read_sql_query('select * from Reviews where score !=3',con)
```

In [12]:
```
raw_data.head()
```

Out[12]:

	Id	ProductId	UserId	ProfileName	\
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
2	3	B000LQOCH0	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      5  1303862400
1                     0                       0      1  1346976000
2                     1                       1      4  1219017600
3                     3                       3      2  1307923200
4                     0                       0      5  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...
```

```
In [13]: def get_sentiment(x):
             if x<3:
                 return "Negative"
             else:
                 return "Positive"


         Sentiment=raw_data.Score.map(get_sentiment)
```

```
In [14]: raw_data['Sentiment']=Sentiment
```

```
In [15]: raw_data.drop(['Id','Score'],inplace=True,axis=1)
```

```
In [16]: raw_data.head()
```

```
Out[16]:     ProductId          UserId                      ProfileName  \
         0  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
         1  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
         2  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
         3  B000UA0QIQ  A395BORC6FGVXV                             Karl
         4  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"

            HelpfulnessNumerator  HelpfulnessDenominator        Time  \
         0                     1                       1  1303862400
         1                     0                       0  1346976000
         2                     1                       1  1219017600
         3                     3                       3  1307923200
         4                     0                       0  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...

  Sentiment
0  Positive
1  Negative
2  Positive
3  Negative
4  Positive
```

In [20]:
```
raw_data.sort_values(by=['Time','UserId'])
```

```
# Since we see lots off duplicates in data side which can me misleading for our futur
```

Out[20]:

	ProductId	UserId	ProfileName	\
138706	0006641040	ACITT7DI6IDDL	shari zychinski	
138683	0006641040	AJ46FKXOVC7NR	Nicholas A Mesiano	
417839	B00004CXX9	AIUWLEQ1ADEG5	Elizabeth Medina	
212472	B00004RYGX	A344SMIA5JECGM	Vincent P. Ross	
346055	B00004CI84	A344SMIA5JECGM	Vincent P. Ross	
417859	B00004CXX9	A344SMIA5JECGM	Vincent P. Ross	
417838	B00004CXX9	AJH6LUC1UT1ON	The Phantom of the Opera	
212533	B00004RYGX	A1048CYU0OV4O8	Judy L. Eans	
346116	B00004CI84	A1048CYU0OV4O8	Judy L. Eans	
417927	B00004CXX9	A1048CYU0OV4O8	Judy L. Eans	
212458	B00004RYGX	A1B2IZU1JLZA6	Wes	
346041	B00004CI84	A1B2IZU1JLZA6	Wes	
417847	B00004CXX9	A1B2IZU1JLZA6	Wes	
70688	B00002N8SM	A32DW342WBJ6BX	Buttersugar	
212558	B00004RYGX	ACJR7EQF9S6FP	Jeremy Robertson	
346141	B00004CI84	ACJR7EQF9S6FP	Jeremy Robertson	
417952	B00004CXX9	ACJR7EQF9S6FP	Jeremy Robertson	
212511	B00004RYGX	A2DEE7F9XKP3ZR	jerome	
346094	B00004CI84	A2DEE7F9XKP3ZR	jerome	
417883	B00004CXX9	A2DEE7F9XKP3ZR	jerome	
1146	B00002Z754	A29Z5PI9BW2PU3	Robbie	
1145	B00002Z754	A3B8RCEI0FXFI6	B G Chase	
121041	B00004RAMX	A5NQLNC6QPGSI	Kim Nason	
138001	B00004S1C5	A1KXONFPU2XQ5K	Stephanie Manley	
138017	B00004S1C6	A1KXONFPU2XQ5K	Stephanie Manley	
212532	B00004RYGX	A1FJOY14X3MUHE	Justin Howard	
346115	B00004CI84	A1FJOY14X3MUHE	Justin Howard	
417926	B00004CXX9	A1FJOY14X3MUHE	Justin Howard	
212519	B00004RYGX	A1GB1Q193DNFGR	Bruce Lee Pullen	
346102	B00004CI84	A1GB1Q193DNFGR	Bruce Lee Pullen	

...	...	...	...
516763	B004XGCWY4	AR0OHXFDJAHG4	Lollie Dot Com "I love my bicycles"
401762	B003ZFUYPI	ARCGKF8PS7JBI	suzy
125540	B001PQOB4O	ARF9K2APJHAH3	L. Abernathy "RN"
235616	B002Y3CSD8	ATL8TGSAEMDV1	Ray
78921	B000NCXHFA	ATMTE0BSP7W6S	Lor "Lor"
104677	B0039JXQ1Y	AUEA2NJHMK9DF	Penny E. Cooke "PMSDEA"
204233	B00028LU40	AUEA2NJHMK9DF	Penny E. Cooke "PMSDEA"
210831	B0001M0ZWA	AUEA2NJHMK9DF	Penny E. Cooke "PMSDEA"
219434	B003ASXKV0	AUEA2NJHMK9DF	Penny E. Cooke "PMSDEA"
244651	B000C08B0E	AUEA2NJHMK9DF	Penny E. Cooke "PMSDEA"
14300	B000255OIG	AUINI96NMGXUI	Kkrys23
86425	B0002DGRPC	AUINI96NMGXUI	Kkrys23
157914	7310172001	AUINI96NMGXUI	Kkrys23
200669	7310172101	AUINI96NMGXUI	Kkrys23
279857	B0002DGRZC	AUINI96NMGXUI	Kkrys23
300207	B0002DGRRA	AUINI96NMGXUI	Kkrys23
365015	B001B4VOQI	AUINI96NMGXUI	Kkrys23
387772	B0002DGRQ6	AUINI96NMGXUI	Kkrys23
491423	B0002DGRSY	AUINI96NMGXUI	Kkrys23
169881	B000P07YA4	AVN9NG27SPISS	Heather Casavant
511105	B0012EYELE	AWQOTHBNJBSVB	Gregory E. Grant "GG"
429396	B000UBD88A	AWRFQYLG7LQKJ	Rocdoc "Roc"
451349	B0013A0QXC	AWRFQYLG7LQKJ	Rocdoc "Roc"
82885	B00866AM2G	AY839W9JQDZM2	Daniella
434639	B001IZIEGS	AYDZMG82XBCPW	M. Reed
55730	B003QNJYXM	AYTSBGA5A3UWI	Imran Ali
280722	B001AS1A4Q	AYTSBGA5A3UWI	Imran Ali
387889	B0029ZAOW8	AYTSBGA5A3UWI	Imran Ali
462720	B003QNLUTI	AYTSBGA5A3UWI	Imran Ali
39050	B0014DXT5A	AZUCLRMHEBUG0	T. HANLEY "reader"

	HelpfulnessNumerator	HelpfulnessDenominator	Time	\
138706	0	0	939340800	
138683	2	2	940809600	
417839	0	0	944092800	
212472	1	2	944438400	
346055	1	2	944438400	
417859	1	2	944438400	
417838	0	0	946857600	
212533	2	2	947376000	
346116	2	2	947376000	
417927	2	2	947376000	
212458	19	23	948240000	
346041	19	23	948240000	
417847	19	23	948240000	
70688	0	0	948672000	
212558	2	3	951523200	

346141	2	3	951523200
417952	2	3	951523200
212511	0	3	959990400
346094	0	3	959990400
417883	0	1	959990400
1146	7	7	961718400
1145	10	10	962236800
121041	7	8	965001600
138001	8	8	965779200
138017	26	28	965779200
212532	2	2	966297600
346115	2	2	966297600
417926	2	2	966297600
212519	5	5	970531200
346102	5	5	970531200
...	...	...	...
516763	0	0	1351209600
401762	0	0	1351209600
125540	0	0	1351209600
235616	0	0	1351209600
78921	0	0	1351209600
104677	0	0	1351209600
204233	0	0	1351209600
210831	0	0	1351209600
219434	0	0	1351209600
244651	0	0	1351209600
14300	0	0	1351209600
86425	0	0	1351209600
157914	0	0	1351209600
200669	0	0	1351209600
279857	0	0	1351209600
300207	0	0	1351209600
365015	0	0	1351209600
387772	0	0	1351209600
491423	0	0	1351209600
169881	0	0	1351209600
511105	0	0	1351209600
429396	0	0	1351209600
451349	0	0	1351209600
82885	0	0	1351209600
434639	0	0	1351209600
55730	0	0	1351209600
280722	0	0	1351209600
387889	0	0	1351209600
462720	0	0	1351209600
39050	0	0	1351209600

Summary \

```
138706                          EVERY book is educational
138683  This whole series is great way to spend time w...
417839                             Entertainingl Funny!
212472                           A modern day fairy tale
346055                           A modern day fairy tale
417859                           A modern day fairy tale
417838                                        FANTASTIC!
212533                                             GREAT
346116                                             GREAT
417927                                             GREAT
212458    WARNING: CLAMSHELL EDITION IS EDITED TV VERSION
346041    WARNING: CLAMSHELL EDITION IS EDITED TV VERSION
417847    WARNING: CLAMSHELL EDITION IS EDITED TV VERSION
70688                             A sure death for flies
212558          Bettlejuice...Bettlejuice...BETTLEJUICE!
346141          Bettlejuice...Bettlejuice...BETTLEJUICE!
417952          Bettlejuice...Bettlejuice...BETTLEJUICE!
212511      Research - Beatlejuice video - French version
346094      Research - Beatlejuice video - French version
417883                                          Research
1146                                       Great Product
1145                      WOW Make your own 'slickers' !
121041                          End your Gopher Problems
138001                                  Very easy to use
138017                                      A must have!
212532  A fresh, original film from master storyteller...
346115  A fresh, original film from master storyteller...
417926  A fresh, original film from master storyteller...
212519       Fabulous Comedic Fanasy Directed by a Master
346102       Fabulous Comedic Fanasy Directed by a Master
...                                                  ...
516763                                          Awesome
401762                                          ultrax3
125540                        Best cornbread mix ever!!!!
235616                               I would buy it again
78921                          Mung Beans for sprouting
104677                                     Like this tea
204233                                     Like this tea
210831                                     Like this tea
219434                                     Like this tea
244651                                     Like this tea
14300                                   Love this faucet
86425                                   Love this faucet
157914                                  Love this faucet
200669                                  Love this faucet
279857                                  Love this faucet
300207                                  Love this faucet
365015                                  Love this faucet
```

387772	Love this faucet
491423	Love this faucet
169881	New Recipe is Awful
511105	Couldn't tell you how it tasted
429396	Not very strong
451349	Not very strong
82885	I love this
434639	Great fresh taste
55730	A God Sent Remedy!!!
280722	A God Sent Remedy!!!
387889	A God Sent Remedy!!!
462720	A God Sent Remedy!!!
39050	ZipFizz liquid energy shot

	Text	Sentiment
138706	this witty little book makes my son laugh at l...	Positive
138683	I can remember seeing the show when it aired o...	Positive
417839	Beetlejuice is a well written movie ... ever...	Positive
212472	A twist of rumplestiskin captured on film, sta...	Positive
346055	A twist of rumplestiskin captured on film, sta...	Positive
417859	A twist of rumplestiskin captured on film, sta...	Positive
417838	Beetlejuice is an excellent and funny movie. K...	Positive
212533	THIS IS ONE MOVIE THAT SHOULD BE IN YOUR MOVIE...	Positive
346116	THIS IS ONE MOVIE THAT SHOULD BE IN YOUR MOVIE...	Positive
417927	THIS IS ONE MOVIE THAT SHOULD BE IN YOUR MOVIE...	Positive
212458	I, myself always enjoyed this movie, it's very...	Negative
346041	I, myself always enjoyed this movie, it's very...	Negative
417847	I, myself always enjoyed this movie, it's very...	Negative
70688	I bought a few of these after my apartment was...	Positive
212558	What happens when you say his name three times...	Positive
346141	What happens when you say his name three times...	Positive
417952	What happens when you say his name three times...	Positive
212511	I'm getting crazy.I'm looking for Beatlejuice ...	Positive
346094	I'm getting crazy.I'm looking for Beatlejuice ...	Positive
417883	I'm getting crazy.<p>Is it really impossible t...	Positive
1146	This was a really good idea and the final prod...	Positive
1145	I just received my shipment and could hardly w...	Positive
121041	I have just recently purchased the Woodstream ...	Positive
138001	This are so much easier to use than the Wilson...	Positive
138017	These are easy to use, they do not make a mess...	Positive
212532	This is such a great film, I don't even know h...	Positive
346115	This is such a great film, I don't even know h...	Positive
417926	This is such a great film, I don't even know h...	Positive
212519	Beetlejuice is an awe-inspiring wonderfully am...	Positive
346102	Beetlejuice is an awe-inspiring wonderfully am...	Positive
...	...	...
516763	I can't find this in the stores and thought I ...	Positive
401762	Pleased with product. Cornstarch and Wondra fo...	Positive

```
125540  Great corn bread mix.Perfect for thanksgiving...  Positive
235616  I decided to try this sauce recently and overa...  Positive
78921   I wanted to feed my chickens, turkeys and goat...  Positive
104677  This tea has a nice flavor although I wish it ...  Positive
204233  This tea has a nice flavor although I wish it ...  Positive
210831  This tea has a nice flavor although I wish it ...  Positive
219434  This tea has a nice flavor although I wish it ...  Positive
244651  This tea has a nice flavor although I wish it ...  Positive
14300   Love this faucet.  My husband had installed th...  Positive
86425   Love this faucet.  My husband had installed th...  Positive
157914  Love this faucet.  My husband had installed th...  Positive
200669  Love this faucet.  My husband had installed th...  Positive
279857  Love this faucet.  My husband had installed th...  Positive
300207  Love this faucet.  My husband had installed th...  Positive
365015  Love this faucet.  My husband had installed th...  Positive
387772  Love this faucet.  My husband had installed th...  Positive
491423  Love this faucet.  My husband had installed th...  Positive
169881  These used to be my favorite animal crackers b...  Negative
511105  The bottle was not sealed, and when I opened t...  Negative
429396  Not as strong as the regular dark coffee.  Dis...  Negative
451349  Not as strong as the regular dark coffee.  Dis...  Negative
82885   It's is in my opinion the best coconut water o...  Positive
434639  These are plump, fresh pistachios with a very ...  Positive
55730   I love this stuff! It's a God sent Remedy for ...  Positive
280722  I love this stuff! It's a God sent Remedy for ...  Positive
387889  I love this stuff! It's a God sent Remedy for ...  Positive
462720  I love this stuff! It's a God sent Remedy for ...  Positive
39050   Have used this for years.  Gives long lasting ...  Positive

[525814 rows x 9 columns]
```

```
In [36]: raw_data.drop_duplicates(inplace=True,subset=['Time','UserId'])

In [37]: raw_data.shape

Out[37]: (328731, 9)

In [38]: raw_data[raw_data.HelpfulnessNumerator>raw_data.HelpfulnessDenominator].index

         # HelpfulnessNumerator must be < HelpfulnessDenominator we have [41159, 59301] index

Out[38]: Int64Index([], dtype='int64')

In [30]: raw_data.drop([41159, 59301],inplace=True)

In [52]: raw_data.reset_index(inplace=True)

In [53]: raw_data.drop('index',inplace=True,axis=1)

In [54]: raw_data.shape
```

```
Out[54]: (328731, 9)

In [55]: filtered_data=raw_data.iloc[:10000,:].copy()   # Filtering only 10K datapoints

In [57]: filtered_data.tail()

Out[57]:        ProductId          UserId                      ProfileName  \
         9995  B005HB4HGU   AX3M0765KGH6P                            Doris
         9996  B001EQ59GE   ADKDYRNQBRTJ9                      Roger Lilly
         9997  B001EQ59GE  A22O6MWQRSNHFX                             didi
         9998  B001EQ59GE  A1H6SBO7R007I8                       A. Reader
         9999  B003HUFLYA  A267V5VTEWEQ1T  Karen A. Carpenter "Dog Lady 6"

               HelpfulnessNumerator  HelpfulnessDenominator        Time  \
         9995                     0                      15  1317686400
         9996                     0                       0  1308096000
         9997                     1                       3  1281139200
         9998                     1                       3  1242172800
         9999                     0                       0  1305504000

                                  Summary  \
         9995                     I like it
         9996                Paprika supplier
         9997            not what I expected
         9998   Dont let the name fool you...
         9999  the perfect commuter dog treat

                                                        Text Sentiment
         9995  i put it in a glass jar for chow watmore ken i...  Positive
         9996  I ordered this product as my previous paprika ...  Positive
         9997  Actually typed in "smoked paprika". This showe...  Negative
         9998  This is regular Paprika you buy in the grocery...  Positive
         9999  Not too dry<br />Not too wet<br />Not too hard...  Positive

In [65]: filtered_data['Sentiment'].value_counts()

Out[65]: Positive    8368
         Negative    1632
         Name: Sentiment, dtype: int64

In [17]: filtered_data.head()

Out[17]:     ProductId          UserId                      ProfileName  \
         0  B001E4KFG0  A3SGXH7AUHU8GW                       delmartian
         1  B00813GRG4  A1D87F6ZCVE5NK                           dll pa
         2  B000LQOCH0   ABXLMWJIXXAIN  Natalia Corres "Natalia Corres"
         3  B000UA0QIQ  A395BORC6FGVXV                             Karl
         4  B006K2ZZ7K  A1UQRSCLF8GW1T    Michael D. Bigham "M. Wassir"
```

```
   HelpfulnessNumerator  HelpfulnessDenominator        Time  \
0                     1                       1  1303862400
1                     0                       0  1346976000
2                     1                       1  1219017600
3                     3                       3  1307923200
4                     0                       0  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...

  Sentiment
0  Positive
1  Negative
2  Positive
3  Negative
4  Positive
```

0.1 Text Preprocessing: Stemming, stop-word removal and Lemmatization.

In the Preprocessing phase we do the following in the order below:-

```
Begin by removing the html tags
Remove any punctuations or limited set of special characters like , or . or # etc.
Check if the word is made up of english letters and is not alpha-numeric
Check to see if the length of the word is greater than 2 (as it was researched that there is no
Convert the word to lowercase
Remove Stopwords
Finally Snowball Stemming the word (it was obsereved to be better than Porter Stemming)
```

```
In [3]: # checking if HTML tags are present

        for rev in filtered_data.Text:
            pattern=re.compile('<*>')
            if pattern.search(rev):
                print (rev)
                break
        # checking if punctuation is present
        for rev in filtered_data.Text:
            pattern=re.compile(r'[?|!|\'|"|.|#|,|)|(|\|/]')
            if pattern.search(rev):
                print (rev)
                break
```

```
I don't know if it's the cactus or the tequila or just the unique combination of ingredients, b
I have bought several of the Vitality canned dog food products and have found them all to be of
```

```
In [7]: # It seems we have HTML tags in our corpus also punctuation is present, which needed t

        def remove_tags(x):
            removed=re.sub('<*>'," ",x)
            return removed
        def remove_punc(x):
            cleaned = re.sub(r'[?|!|\'|"|#]',r'',x)
            cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',x)
            return cleaned

In [8]: # Stopword check

        sno=nltk.stem.SnowballStemmer(language='english')

In [2]: nltk.download('stopwords')

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\andy\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.


Out[2]: True

In [9]: stop=set(stopwords.words('english'))
        print(stop)

{'re', 'nor', 'y', 'doing', 'my', 'because', 'against', 'more', 'such', 'haven', 'was', "hasn't


In [10]: # Intializing Snowball stemmer

         sno=nltk.stem.SnowballStemmer('english')
         sno.stem('yummy')

Out[10]: 'yummi'

In [11]: #code copied from main ipynb and modified to create cleaned text:

         i=0
         str1=' '
         final_string=[]
         all_positive_words=[] # store words from +ve reviews here
         all_negative_words=[] # store words from -ve reviews here.
         s=''
         for sent in filtered_data.Text:
             filtered_sentence=[]
             #print(sent);
```

```
            sent=remove_tags(sent) # remove HTMl tags
            for w in sent.split():
                for cleaned_words in remove_punc(w).split():
                    if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                        if(cleaned_words.lower() not in stop):
                            s=(sno.stem(cleaned_words.lower())).encode('utf8')
                            filtered_sentence.append(s)
                            if filtered_data['Sentiment'].values[i] == 'Positive':
                                all_positive_words.append(s) #list of all words used to descr
                            if filtered_data['Sentiment'].values[i] == 'Negative':
                                all_negative_words.append(s) #list of all words used to descr
                        else:
                            continue
                    else:
                        continue
            #print(filtered_sentence)
            str1 = b" ".join(filtered_sentence) #final string of cleaned words
            #print("*************************************************************")

            final_string.append(str1)
            i+=1
```

```
In [12]: filtered_data['CleanedText']=final_string #adding a column of CleanedText which displ
         filtered_data['CleanedText']=filtered_data['CleanedText'].str.decode("utf-8")
```

```
In [32]: filtered_data.head()
```

Out[32]:

	ProductId	UserId	ProfileName	\
0	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	
1	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
2	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	
3	B000UA0QIQ	A395BORC6FGVXV	Karl	
4	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	

	HelpfulnessNumerator	HelpfulnessDenominator	Time	\
0	1	1	1303862400	
1	0	0	1346976000	
2	1	1	1219017600	
3	3	3	1307923200	
4	0	0	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...	

```
  Sentiment                                        CleanedText
0  Positive  bought sever vital can dog food product found ...
1  Negative  product arriv label jumbo salt peanut peanut a...
2  Positive  confect around centuri light pillowi citrus ge...
3  Negative  look secret ingredi robitussin believ found go...
4  Positive  great taffi great price wide assort yummi taff...
```

```
In [34]: # store final table into an SQlLite table for future.
         conn = sqlite3.connect('filtered_data.sqlite')
         c=conn.cursor()
         conn.text_factory = str
         filtered_data.to_sql('Reviews', conn,  schema=None, if_exists='replace', index=True,
```

```
In [13]: output = open('data.pkl', 'wb')

         pickle.dump(filtered_data, output)


         output.close()
```

```
In [2]: pkl_file = open('data.pkl', 'rb')
        filtered_data = pickle.load(pkl_file)
        #pprint.pprint(filtered_data)
        pkl_file.close()
```

Filtering 2000 Positive and 2000 Negaitive review since more data points are creating :MemoryError

```
In [4]: filter_pos=filtered_data[filtered_data.Sentiment=="Positive"]
        filter_pos=filtered_data.iloc[:2000,:]
        filter_neg=filtered_data[filtered_data.Sentiment=="Negative"]
        filter_neg=filtered_data.iloc[:2000,:]
```

```
In [10]: filtered_4000=pd.concat([filter_pos, filter_neg], ignore_index=True)
```

```
In [17]: filtered_4000.reset_index(inplace=True)
```

0.1.1 Bag Of Words

```
In [18]: # BOW=Bi grams


         count_vector=CountVectorizer(ngram_range=(1,2))
         filtered_data_count=count_vector.fit_transform(filtered_4000.CleanedText)
         print("the type of count vectorizer ",type(filtered_data_count))
         print("the shape of out text BOW vectorizer ",filtered_data_count.get_shape())
         print("the number of unique words ", filtered_data_count.get_shape()[1])
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (4000, 53711)
the number of unique words  53711
```

```
In [19]: filtered_data_count=filtered_data_count.toarray()
```

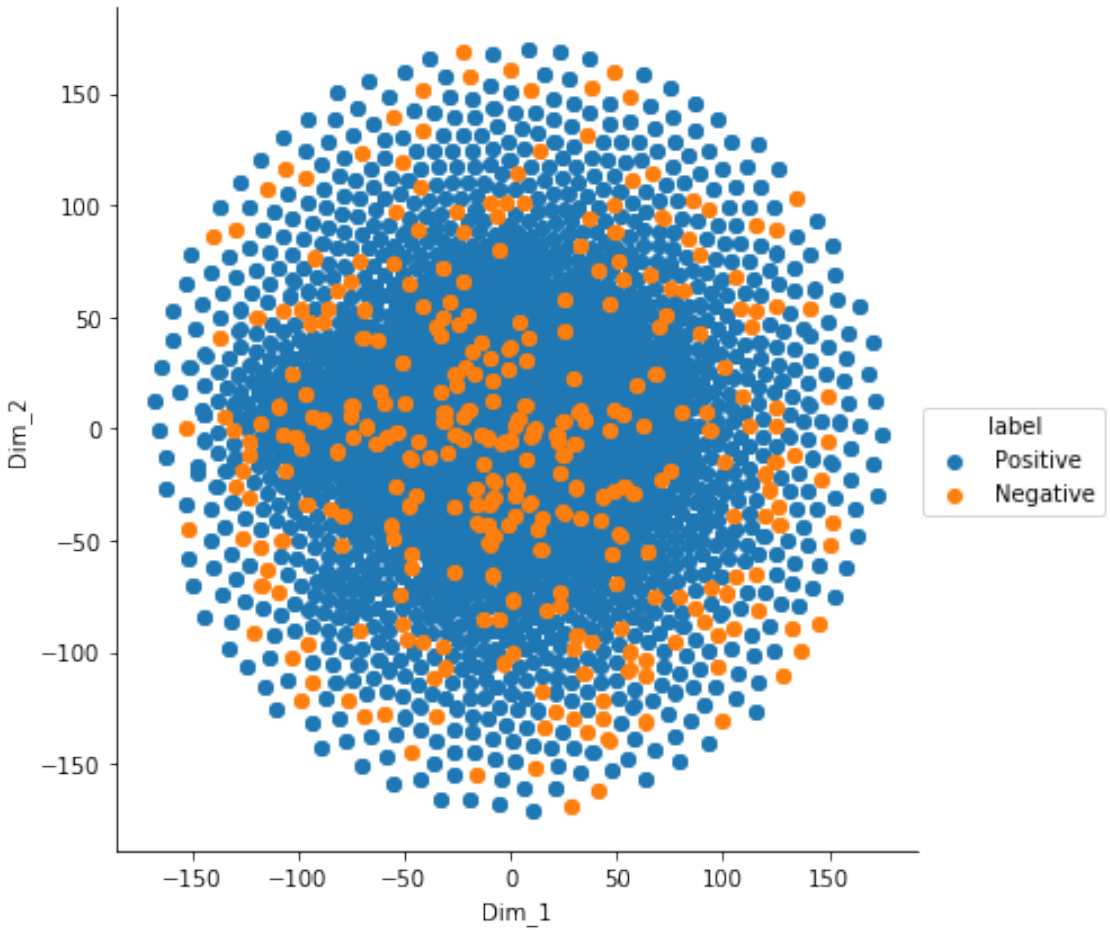
```
In [21]: filtered_data_count.shape
```

```
Out[21]: (4000, 53711)
```

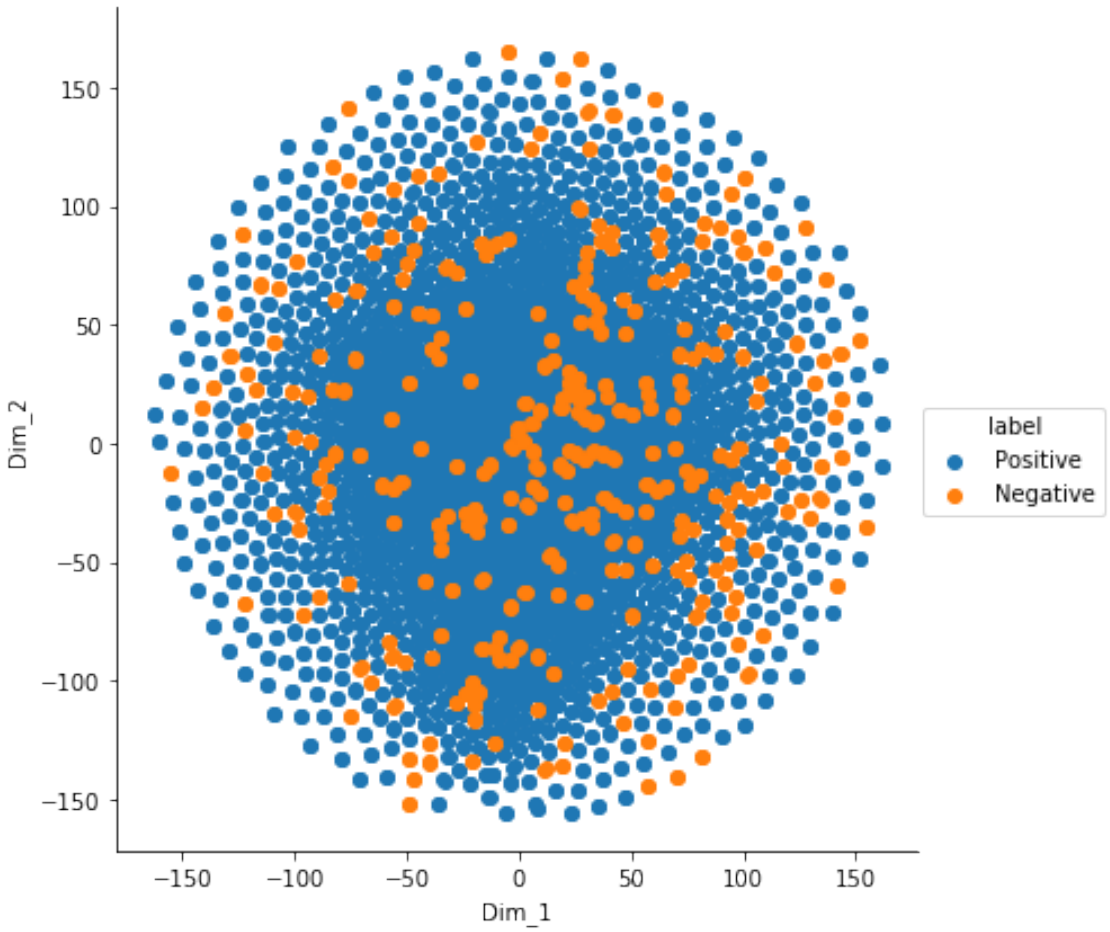
0.1.2 Creating function for applying and plotting TSNE

```
In [33]: def apply_and_plot_tsne(data,label_column,perplexity=30,iterataion=1000):

             model = TSNE(n_components=2, random_state=50,perplexity=perplexity,n_iter=iterata
         # configuring the parameteres
         # the number of components = 2
         # default perplexity = 30
         # default learning rate = 200
         # default Maximum number of iterations for the optimization = 1000

             tsne_data = model.fit_transform(data)


         # creating a new data frame which help us in ploting the result data
             tsne_data = np.vstack((tsne_data.T, label_column)).T
             tsne_df = pd.DataFrame(data=tsne_data, columns=("Dim_1", "Dim_2", "label"))

         # Ploting the result of tsne
             sns.FacetGrid(tsne_df, hue="label", size=6).map(plt.scatter, 'Dim_1', 'Dim_2').ad
             plt.show()
```

```
In [34]: apply_and_plot_tsne(filtered_data_count,filtered_4000.Sentiment,50)
```



In [40]: `apply_and_plot_tsne(filtered_data_count,filtered_4000.Sentiment,70)`



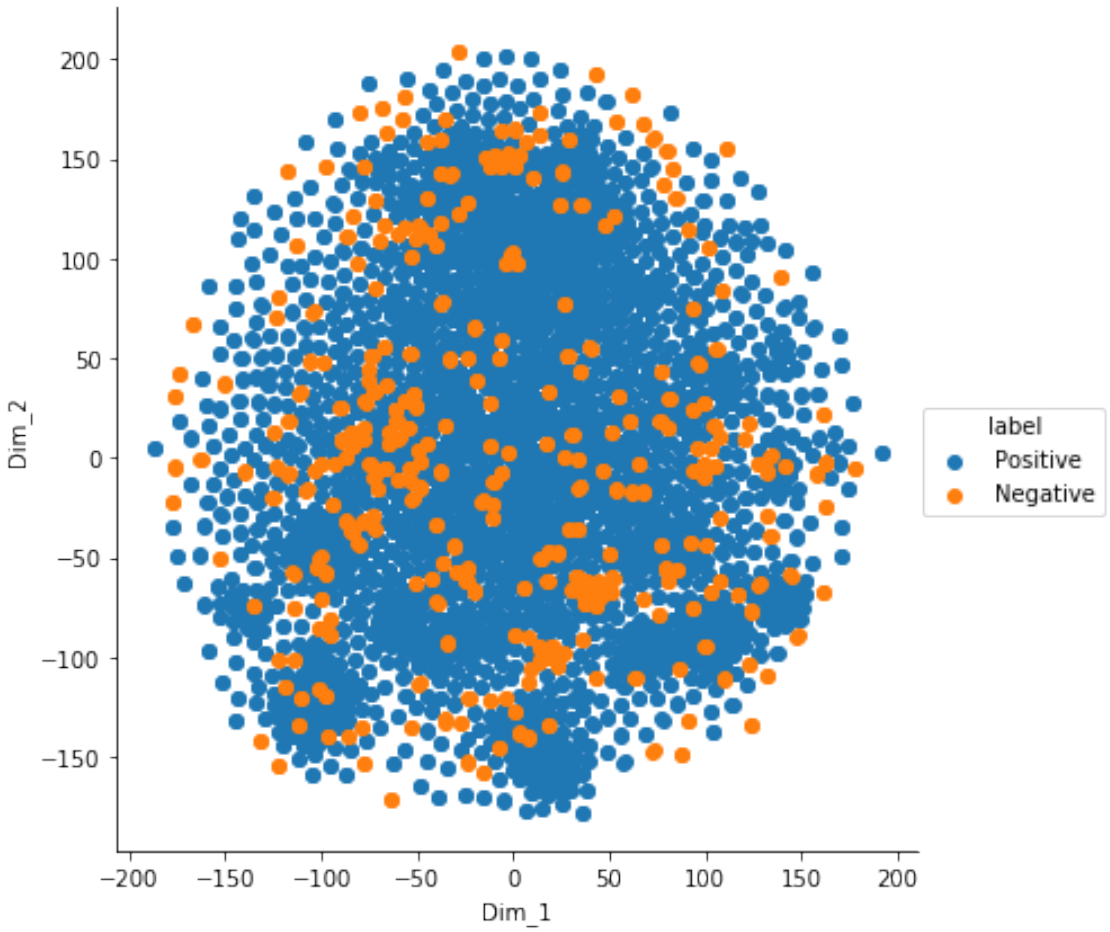
0.1.3 TF-IDF intialization and dimension creation

```
In [37]: tf_idf_vect = TfidfVectorizer(ngram_range=(1,2))
         filtered_tf_idf = tf_idf_vect.fit_transform(filtered_4000.CleanedText)
         print("the type of count vectorizer ",type(filtered_tf_idf))
         print("the shape of out TF-IDF vectorizer ",filtered_tf_idf.get_shape())
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out TF-IDF vectorizer  (4000, 53711)
```

```
In [38]: filtered_tf_idf=filtered_tf_idf.toarray()
```

```
In [39]: apply_and_plot_tsne(filtered_tf_idf,filtered_4000.Sentiment)
```



0.1.4 Wieghted Word2Vec

```
In [41]: list_of_sent=[]
         for sent in filtered_4000.CleanedText:
             list_of_sent.append(sent.split())

In [48]: print(len(list_of_sent))
         print(list_of_sent[0])

4000
['bought', 'sever', 'vital', 'can', 'dog', 'food', 'product', 'found', 'good', 'qualiti', 'proc

In [53]: w2v_model=Word2Vec(list_of_sent,min_count=5,size=50, workers=4)
         w2v_words = list(w2v_model.wv.vocab)
         # average Word2Vec
         # compute average word2vec for each review.
         sent_vectors = []; # the avg-w2v for each sentence/review is stored in this list
```

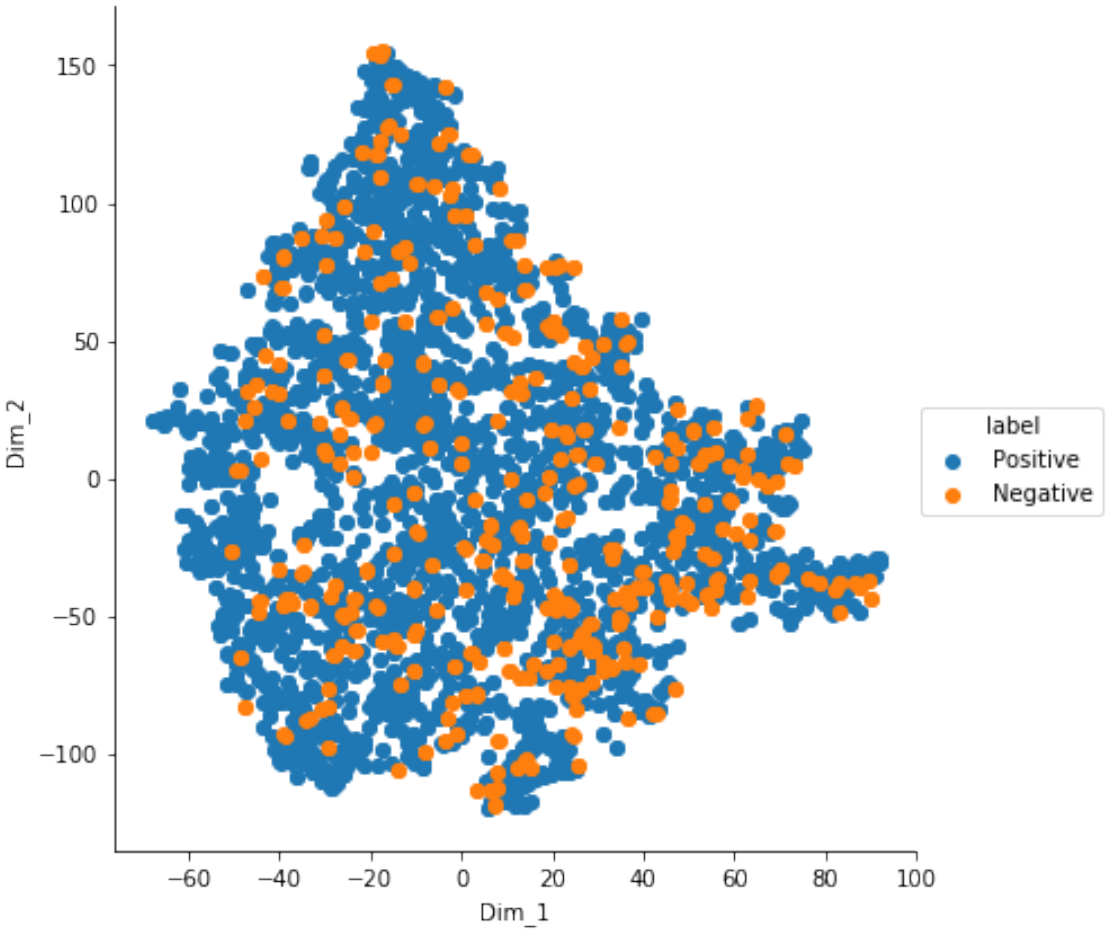
```
    for sent in list_of_sent: # for each review/sentence
        sent_vec = np.zeros(50) # as word vectors are of zero length
        cnt_words =0; # num of words with a valid vector in the sentence/review
        for word in sent: # for each word in a review/sentence
            if word in w2v_words:
                vec = w2v_model.wv[word]
                sent_vec += vec
                cnt_words += 1
        if cnt_words != 0:
            sent_vec /= cnt_words
        sent_vectors.append(sent_vec)
    print(len(sent_vectors))
    print(len(sent_vectors[0]))
```

```
4000
50
```

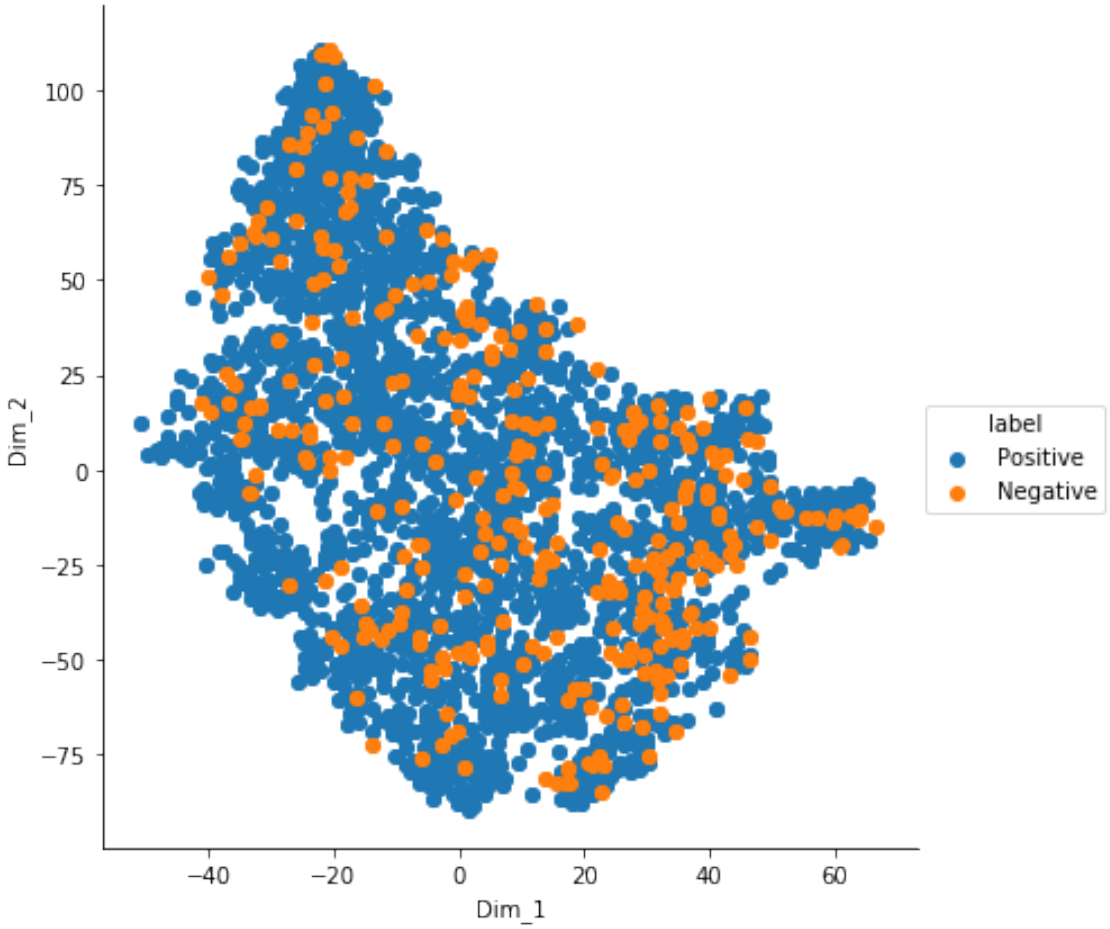
In [55]:
```
type(sent_vectors)
```

Out[55]: list

In [57]:
```
apply_and_plot_tsne(sent_vectors,filtered_4000.Sentiment,50)
```



In [58]: apply_and_plot_tsne(sent_vectors,filtered_4000.Sentiment,100)



0.1.5 Wieghted TF-IDF

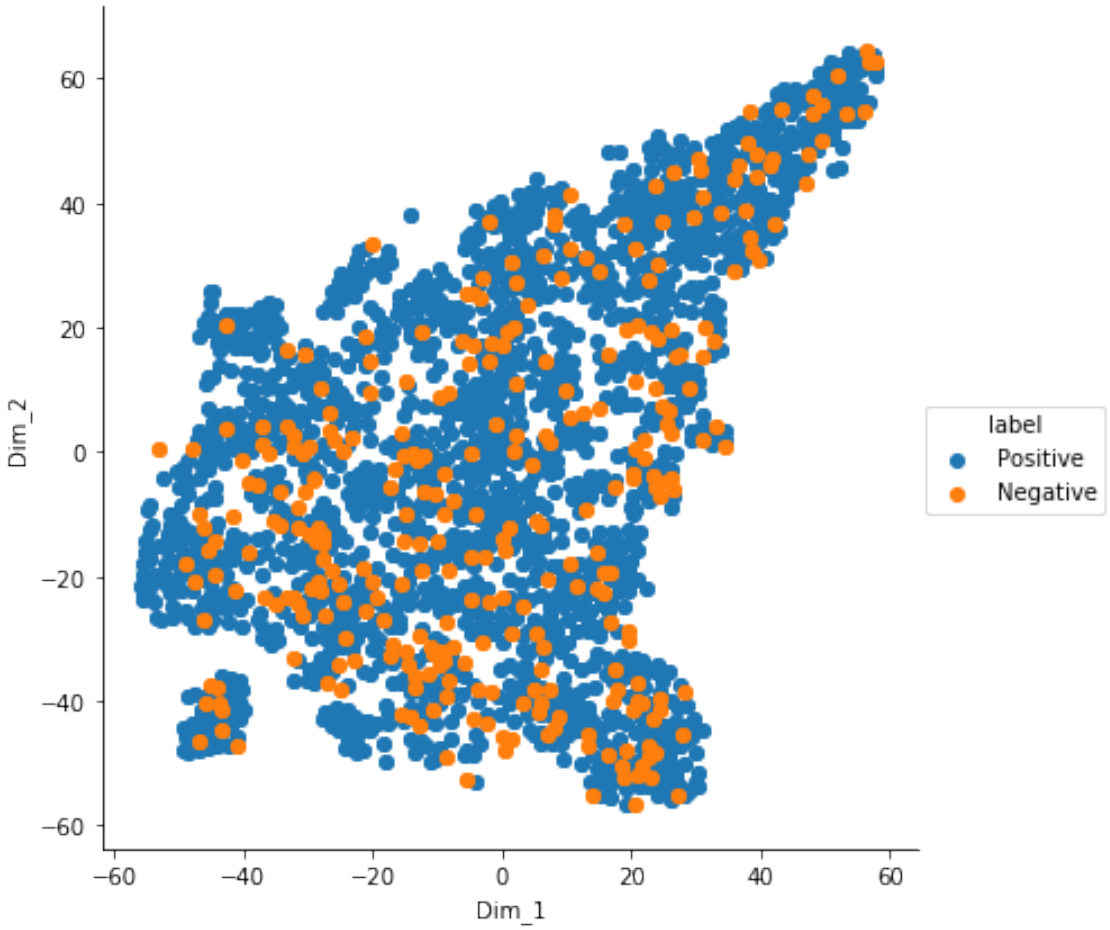
```
In [60]: tfidf_feat = tf_idf_vect.get_feature_names() # tfidf words/col-names
         # final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfidf

         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
                 if word in w2v_words:
                     vec = w2v_model.wv[word]
```

```
                # obtain the tf_idfidf of a word in a sentence/review
                tf_idf = filtered_tf_idf[row, tfidf_feat.index(word)]
                sent_vec += (vec * tf_idf)
                weight_sum += tf_idf
        if weight_sum != 0:
            sent_vec /= weight_sum
        tfidf_sent_vectors.append(sent_vec)
        row += 1
```

In [65]:
```
type(tfidf_sent_vectors)
```

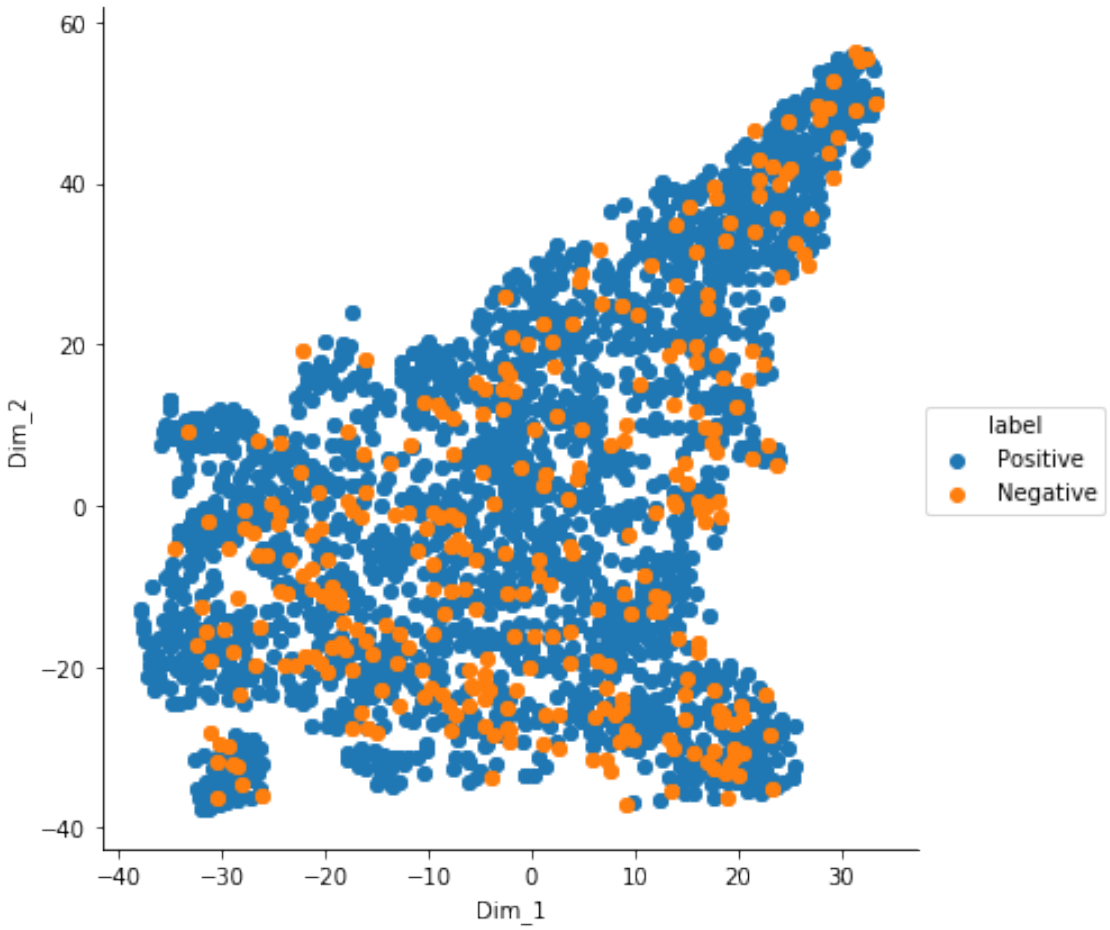
Out[65]: list

In [66]:
```
apply_and_plot_tsne(tfidf_sent_vectors,filtered_4000.Sentiment,100)
```



In [67]:
```
apply_and_plot_tsne(tfidf_sent_vectors,filtered_4000.Sentiment,50)
```

In [68]: `apply_and_plot_tsne(tfidf_sent_vectors,filtered_4000.Sentiment,150)`



0.2 Tried for few values of perplexity but couldn't tried for more as its taking huge processing time for each run.