Applying the t-SNE on Amazon Food review datasets.

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews 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: Id ProductId - unique identifier for the product UserId - unqiue identifier for the user ProfileName HelpfulnessNumerator - number of users who found the review helpful HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not Score - rating between 1 and 5 Time - timestamp for the review Summary - brief summary of the review Text - text of the review

Note:I have considered 8000 sample datapoints to do this analysis.

Note2: Before starting each line i have given explanation of what each code line meaning. This will make reader understand the work better.

Objective:

```
Converting the review text into vector using technique like BOW(Bag of Words,Average
Word2vec,TF-IDF weighted word2vec)

TSE to reduce multiple dimension to 2 dimensions.

Plotting the data using Seaborn.
```

Importing the warning module to supress the warning generated while building the model.

In [38]:

```
import warnings
warnings.filterwarnings("ignore")
```

Importing all the neccessary packages.

In [39]:

```
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
import nltk #newly introduced#
import string #newly introduced#
import sqlite3 #newly introduced#
from sklearn.feature_extraction.text import TfidfTransformer #newly introduced#
from sklearn.feature_extraction.text import TfidfVectorizer #newly introduced#
from sklearn.feature_extraction.text import CountVectorizer #newly introduced#
from sklearn.metrics import confusion_matrix #newly introduced#
from sklearn import metrics #newly introduced#
from sklearn.metrics import roc_curve, auc #newly introduced#
from nltk.stem.porter import PorterStemmer #newly introduced#
from nltk.corpus import stopwords #newly introduced#
from nltk.stem import PorterStemmer #newly introduced#
from nltk.stem.wordnet import WordNetLemmatizer #newly introduced#
from gensim.models import Word2Vec #newly introduced#
```

```
from gensim.models import Word2Vec #newly introduced#
from gensim.models import KeyedVectors #newly introduced#
import pickle #newly introduced#
from tqdm import tqdm #newly introduced#
import os #newly introduced#
```

Loading the data

connecting with the database using connect function in sqlite.

In [40]:

```
connect_database=sqlite3.connect("database.sqlite")
```

the database name is Reviews and it has column Score and then applying a simple SQL command to select everything where score is not equal to 3 and size of the database is limited to 8000

In [41]:

```
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 LIMIT 8000""",
connect_database)
```

displaying the filtered data.

In [42]:

```
filtered_data
```

Out[42]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Tim
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	5	130386240
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	1	134697600
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	4	121901760
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	2	130792320
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	5	135077760
5	6	B006K2ZZ7K	ADT0SRK1MGOEU	Twoapennything	0	0	4	134205120
6	7	B006K2ZZ7K	A1SP2KVKFXXRU1	David C. Sullivan	0	0	5	134015040

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Tim
7	8	B006K2ZZ7K	A3JRGQVEQN31IQ	Pamela G. Williams	0	0	5	133600320
8	9	B000E7L2R4	A1MZYO9TZK0BBI	R. James	1	1	5	132200640
9	10	B00171APVA	A21BT40VZCCYT4	Carol A. Reed	0	0	5	135120960
10	11	B0001PB9FE	A3HDKO7OW0QNK4	Canadian Fan	1	1	5	110782080
11	12	B0009XLVG0	A2725IB4YY9JEB	A Poeng "SparkyGoHome"	4	4	5	128286720
12	13	B0009XLVG0	A327PCT23YH90	LT	1	1	1	133954560
13	14	B001GVISJM	A18ECVX2RJ7HUE	willie "roadie"	2	2	4	128891520
14	15	B001GVISJM	A2MUGFV2TDQ47K	Lynrie "Oh HELL no"	4	5	5	126835200
15	16	B001GVISJM	A1CZX3CP8IKQIJ	Brian A. Lee	4	5	5	126204480
16	17	B001GVISJM	A3KLWF6WQ5BNYO	Erica Neathery	0	0	2	134809920
17	18	B001GVISJM	AFKW14U97Z6QO	Becca	0	0	5	134507520
18	19	B001GVISJM	A2A9X58G2GTBLP	Wolfee1	0	0	5	132459840
19	20	B001GVISJM	A3IV7CL2C13K2U	Greg	0	0	5	131803200
20	21	B001GVISJM	A1WO0KGLPR5PV6	mom2emma	0	0	5	131345280
21	22	B001GVISJM	AZOF9E17RGZH8	Tammy Anderson	0	0	5	130896000
22	23	B001GVISJM	ARYVQL4N737A1	Charles Brown	0	0	5	130489920

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Tim
23	24	B001GVISJM	AJ613OLZZUG7V	Mare's	0	0	5	130446720
24	25	B001GVISJM	A22P2J09NJ9HKE	S. Cabanaugh "jilly pepper"	0	0	5	129548160
25	26	B001GVISJM	A3FONPR03H3PJS	Deborah S. Linzer "Cat Lady"	0	0	5	128831040
26	27	B001GVISJM	A3RXAU2N8KV45G	lady21	0	1	1	133263360
27	28	B001GVISJM	AAAS38B98HMIK	Heather Dube	0	1	4	133185600
28	29	B00144C10S	A2F4LZVGFLD1OB	DaisyH	0	0	5	133885440
29	30	B0001PB9FY	A3HDKO7OW0QNK4	Canadian Fan	1	1	5	110782080
...	...	...	...	...	...	...	...	...
7970	8721	B003VXFK44	A16K66OVVFCCBD	Miss Ellie	2	2	5	128468160
7971	8722	B003VXFK44	A154WMWOARJHLI	Gary Rosenfeld	2	2	5	128468160
7972	8723	B003VXFK44	A1CSRU8Z5KIPV6	emily p jenkins	2	2	2	128442240
7973	8725	B003VXFK44	A1GDFWGKJARI4M	Jaykid007	2	2	5	128347200
7974	8726	B003VXFK44	A1SUGP9Q67CP28	Judith Ann Horne	1	1	4	134671680
7975	8727	B003VXFK44	A1QHJCHVHQAP4I	heathbc888	1	1	5	134144640
7976	8728	B003VXFK44	A3RQYPFPM58CKP	Norah Rice	1	1	4	133945920
7977	8729	B003VXFK44	A19N301CQ8IWW9	Nikki N.	1	1	5	133695360
7978	8730	B003VXFK44	A33BX5D4DKN65U	Mr. Newman	1	1	4	133375680

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Tim
7979	8731	B003VXFK44	A1MHY69POZ8GCK	Lily	1	1	5	133375680
7980	8732	B003VXFK44	A3LZCRW9NU0327	Tinley Park Shopper	1	1	5	133358400
7981	8734	B003VXFK44	A2K04TXWAV3PZ3	William D. Dillon	1	1	5	133263360
7982	8735	B003VXFK44	AQQLWCMRNDFGI	Steven A. Peterson	1	1	4	132287040
7983	8738	B003VXFK44	A3R7R8ARVN2P3D	A. Kehoe Jr.	1	1	4	131302080
7984	8739	B003VXFK44	A1XAUZ08A5EXJA	Ming	1	1	4	130368960
7985	8740	B000LKZ84C	AKYYV4PLEJRJJ	SmarterThanYouThink	27	29	4	121556160
7986	8741	B000LKZ84C	A2UDUJG17I8S1	RunVeg	17	18	5	125668800
7987	8742	B000LKZ84C	A16KL4A6Z8GVXR	Stephen Gereb	7	7	5	132770880
7988	8743	B000LKZ84C	AROZR86IK0KO2	Soe Mi Kyeong	9	10	5	123854400
7989	8744	B000LKZ84C	A2VDQUOEN5SYKD	Coconut "Unconstipated Carbivore Loiterer"	6	6	5	133807680
7990	8745	B000LKZ84C	A18Q1Z01K679WR	Alan Truly	3	3	5	133816320
7991	8746	B000LKZ84C	A2B3DDF9Q5VEUA	Samantha	3	3	4	132295680
7992	8747	B000LKZ84C	AAY43PCKCVONT	f.	3	3	4	130904640
7993	8748	B000LKZ84C	A1WQP7LCVQK3JZ	zmkr788	2	2	5	130256640
7994	8749	B000LKZ84C	A2WQ4FT0CMDSUI	Profane Poet	2	2	4	129211200

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Tim
7995	8750	B000LKZ84C	AX668BUQRHI7C	stephie	1	1	5	134576640
7996	8751	B000LKZ84C	APGQCBO10LZTH	LZ	1	1	5	133721280
7997	8752	B000LKZ84C	A6KL17KKN0A5L	K. Harper	1	1	5	130818240
7998	8753	B000LKZ84C	A293N34D1VDE3	MJL	2	3	5	130248000
7999	8754	B000LKZ84C	A3J30T6XOU0BWS	Marcia Bicknell	2	3	5	126083520

8000 rows × 10 columns

To know the number of rows and column in the sample dataset.

In [43]:

```
filtered_data.shape
```

Out[43]:

```
(8000, 10)
```

A very simple userdefined function to classify x is 0 when the x value is less than 3 and x is 1 when x is more than 3

A important point to consider that Score which contain only value less than 3 and more than 3 but not equal to 3. As score with value 3 will not classify a review as positive or negative.

In [44]:

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

Storing the Score column of the filtered_data dataframe in actualScore

In [45]:

```
actualScore=filtered_data["Score"]
```

checking the shape of column actualScore

checking the shape of column actualScore

In [46]:

```
actualScore.shape
```

Out[46]:

```
(8000,)
```

applying the partition function using map function to actualScore dataframe.

In [47]:

```
positiveNegative=actualScore.map(partition)
```

After applying the partition function Score with more than 3 is assigned to value 1 and score with less than 3 is assigned to value 0.

In [48]:

```
positiveNegative
```

Out[48]:

```
0       positive
1       negative
2       positive
3       negative
4       positive
5       positive
6       positive
7       positive
8       positive
9       positive
10      positive
11      positive
12      negative
13      positive
14      positive
15      positive
16      negative
17      positive
18      positive
19      positive
20      positive
21      positive
22      positive
23      positive
24      positive
25      positive
26      negative
27      positive
28      positive
29      positive
          ...
7970    positive
7971    positive
7972    negative
7973    positive
7974    positive
7975    positive
7976    positive
7977    positive
7978    positive
7979    positive
7980    positive
```

```
7981    positive
7982    positive
7983    positive
7984    positive
7985    positive
7986    positive
7987    positive
7988    positive
7989    positive
7990    positive
7991    positive
7992    positive
7993    positive
7994    positive
7995    positive
7996    positive
7997    positive
7998    positive
7999    positive
Name: Score, Length: 8000, dtype: object
```

pasting the positiveNegative value in the Score column of filtered_data.

In [49]:

```
filtered_data["Score"]=positiveNegative
```

displaying the 5 datapoints of filtered_data after appending pasting the positiveNegative value in the Score column of filtered_data.

In [50]:

```
filtered_data.head()
```

Out[50]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	positive	1303862400	Good Quality Dog Food
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	negative	1346976000	Not as Advertised
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	positive	1219017600	"Delight" says it all
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	negative	1307923200	Cough Medicine
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	positive	1350777600	Great taffy

Displaying the shape of sample dataset.

In [51]:

```
filtered_data.shape
```

Out[51]:

```
(8000, 10)
```

Preprocessing and cleaning the data

Sorting the data as per ProductId. By default it will be arranged in ascending order.

In [52]:

```
sorted_data=filtered_data.sort_values(by='ProductId', axis=0, ascending=True, inplace=False, kind='
quicksort', na_position='last')
```

displaying the last 50 datpoints of the sorted sample dataset.

In [53]:

```
sorted_data.tail(50)
```

Out[53]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Ti
2226	2420	B0089SPDUW	A107SVKYGPGBPP	Angle Side Side	0	0	positive	1335398
2213	2406	B0089SPDUW	A2M98NPVGQX2GK	Camille	0	0	positive	1349827
2227	2421	B0089SPDUW	A3JXSK9WWJU7RT	R. Balakir	0	0	positive	1335139
2212	2405	B0089SPDUW	AA7STIIEST2HW	James Glenny	0	0	positive	1350604
2211	2404	B0089SPDUW	A311QDUHEORR5P	Patrick J Fitzgerald	0	0	positive	1350604
2210	2403	B0089SPDUW	A2LVFLPZ6EHQWR	Donald C. Beck	1	1	positive	1320451
2209	2402	B0089SPDUW	A2TJG4N8LNJW23	Blythe Dresser	1	1	positive	1327104
2208	2401	B0089SPDUW	A2LQ494BOMR72G	D. Walker III	1	1	positive	1330128
2225	2418	B0089SPDUW	A8OHJUH0WSPVI	Jean Strahan "SuperNonna"	0	0	positive	1336003
5423	5874	B008AHJZTM	AOQ2IB802NXAQ	Irishlawlass	0	1	negative	1346889
5422	5873	B008AHJZTM	AKIMXHXC7X8	Tracy Pace	0	0	positive	1348358

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Ti
678	730	B008BEGP9W	A1MJ0MF2OFIU4D	Chris	0	0	positive	1340496
677	729	B008BEGP9W	A2L0EIYNODCZ9U	good dog	0	0	positive	1346371
5915	6405	B008D8A9P2	AZB0JPRWPUSZ7	M. Ariani	0	0	positive	1349395
6968	7616	B008DGFTU4	A38VU3OH6JMAUK	Heather S.	0	0	positive	1350864
5585	6046	B008EE2PNO	A35EIBS4BC1JT2	medicinewoman	0	0	positive	1350691
3663	3980	B008EMA3AS	ABZL1Q5Y7EK9A	M. Heath "Techno guy"	1	1	positive	1346198
3664	3981	B008EMA3AS	A199USG4MU1NWI	rtb521	0	0	positive	1350000
3662	3979	B008EMA3AS	A1TH877IIP7UHX	Karina Feld "karina1999"	2	2	positive	1344729
3580	3891	B008G01KM8	A1H1DVTIHXV3YE	Ryan D. Crompton	0	0	positive	1348531
5137	5571	B008G37TFM	AL4E8JTTW1JL2	Foodie	0	0	positive	1350000
1108	1201	B008L19ZQ0	A1QQ60DX82BE3W	Lamb	1	1	positive	1327795
1107	1200	B008L19ZQ0	A306PAX3GWF5KV	Season Balik	2	2	positive	1323993
1106	1199	B008L19ZQ0	A1N5M6H4GKNBG6	Shanna	2	2	positive	1340409
1109	1202	B008L19ZQ0	A2P19WZ6WD2OZK	Brock	0	0	positive	1346457
1110	1203	B008L19ZQ0	A2BN0B0CJQI3KP	RP	0	0	positive	1334620
1232	1332	B008MMLXEK	A38MF3LIFBPX51	Jordan	0	0	positive	1348617
6021	6519	B008OV8RE8	A3HPCRD9RX351S	Spudman	2	2	positive	1346112
6022	6520	B008OV8RE8	A3HPCRD9RX351S	Spudman	1	1	positive	1345248
4714	5116	B008QXKU4O	A1NJXOUMOIY96X	nanahass	0	0	positive	1350172
5429	5881	B008YA1TNA	A2EBMYVJAEIK75	Hobo Jan	0	0	positive	1348876
2013	2197	B008YAXFWI	AY12DBB0U420B	Gary Peterson	0	0	positive	1346630
5008	5436	B008YGWIZM	AC1G9PEVXK8JJ	Yitbos78	1	1	positive	1346025
5009	5437	B008YGWIZM	A2Q68SJQ4GQN3F	Alissa N. Mattson "Mattsonsonthemove"	0	0	positive	1346716
5010	5438	B008YGWIZM	A33947M1Y587GX	Noodles	0	0	positive	1346025
3567	3878	B009166ECC	A2A5SQE8EEDLLD	Kurt	1	1	negative	1348012
5011	5439	B0092X7B5S	AKD1SD4I503SH	Judith E. Golden	0	0	positive	1278374
5012	5440	B0092X7B5S	A3IE8OGKQ0OCTE	Clifton Watson "Jus' a simple man. Nothin'	3	10	positive	1211760

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Ti
3271	3564	B0092XAMDQ	AM0VRGASSST2K	Ashley	1	1	positive	13469760
220	240	B0093NIWVO	A1JT114SOITFFO	Dan & Eileen	0	0	positive	1350691
6056	6556	B00959DMWK	A14HCVAESP0PPF	lucy burany	0	0	positive	13474080
4118	4462	B0096E5196	A2REBFO0U8B6Q1	Topkat2 "topkat2"	0	0	positive	1350604
4117	4461	B0096E5196	AF6MP20RXGN2J	NORMS09	0	0	positive	1350777
713	768	B009HINRX8	A2CAZG1CQ8BQI5	Patricia J. Nohalty	0	0	positive	1337212
712	766	B009HINRX8	A39BLB42U7M6BD	James Brooks	0	0	positive	1344643
711	765	B009HINRX8	A1OEL4UZT3KKI4	coffee drinker in PA "coffee drinker in PA"	0	0	positive	1344988
710	764	B009HINRX8	ADDBLG0CFY9AI	S.A.D.	1	1	positive	1326758
709	763	B009HINRX8	A3N9477PUE6WMR	patc477	4	4	positive	1323302
1362	1478	B009UOFU20	AJVB004EB0MVK	D. Christofferson	0	0	negative	1345852
5259	5703	B009WSNWC4	AMP7K1O84DH1T	ESTY	0	0	positive	1351209

If the UserId,ProfileName,Text,Time are same then remove the duplicates keeping the first one.

Inplace=False means after droping the duplicates returned the unique datapoints.

In [54]:

```
final_afterRemovingDuplicating=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Text","T
ime"},keep="first",inplace=False)
```

clearly we able to remove 25 duplicate datapoints out of 8000 datapoints.

In [55]:

```
final_afterRemovingDuplicating.shape
```

Out[55]:

```
(7975, 10)
```

No of datapoints in final_afterRemovingDuplicating divided by filtered_data multiplied by 100 will give the infomation about percentage of data retention.

In [56]:

```
((final_afterRemovingDuplicating["Id"].size)/(filtered_data["Id"].size)*100)
```

Out[56]:

```
99.6875
```

Displaying the first 20 datapoints after removing the duplicates.

In [57]:

```
final_afterRemovingDuplicating.head(20)
```

Out[57]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Sum
2546	2774	B00002NCJC	A196AJHU9EASJN	Alex Chaffee	0	0	positive	1282953600	bı
2547	2775	B00002NCJC	A13RRPGE79XFFH	reader48	0	0	positive	1281052800	Be
1146	1245	B00002Z754	A29Z5PI9BW2PU3	Robbie	7	7	positive	961718400	Pr
1145	1244	B00002Z754	A3B8RCEI0FXFI6	B G Chase	10	10	positive	962236800	\ you 'slicl
2942	3204	B000084DVR	A1UGDJP1ZJWVPF	T. Moore "thoughtful reader"	1	1	positive	1177977600	
2941	3203	B000084DVR	A3DKGXWUEP1AI2	Glenna E. Bauer "Puppy Mum"	3	3	positive	1163030400	Pre C F
1071	1161	B000084E1U	A3DH85EYHW4AQH	Eric Hochman	1	1	positive	1140739200	Cat:
5905	6395	B000084EK5	A1Z54EM24Y40LL	c2	1	1	positive	1090972800	F fav look :
5906	6396	B000084EK6	A1Z54EM24Y40LL	c2	0	0	positive	1091059200	
5907	6397	B000084EK7	A1Z54EM24Y40LL	c2	0	0	positive	1090972800	Wha
5897	6386	B000084EK9	A1Z54EM24Y40LL	c2	0	0	negative	1090972800	This is
5896	6385	B000084EK9	AUQIKXJAWMOK5	Desert Rat	0	0	positive	1232496000	Ou lov
5895	6384	B000084EK9	A2OVA909VD90P6	Caryn Trungale Sova	0	0	positive	1325808000	Fou lı
5885	6373	B000084EKA	A1Z54EM24Y40LL	c2	0	0	positive	1090972800	No t

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Sum
5886	6374	B000084EKB	A1Z54EM24Y40LL	c2	0	0	positive	1091059200	
5887	6375	B000084EKC	A1Z54EM24Y40LL	c2	1	1	positive	1090972800	food
5888	6376	B000084EKD	A1Z54EM24Y40LL	c2	1	1	negative	1090972800	N imp
5893	6381	B000084EKG	A1Z54EM24Y40LL	c2	2	2	positive	1090972800	Awe
5894	6382	B000084EKG	A1BU6BSOO8WE5T	Jennifer	1	1	positive	1196035200	My fav
5883	6370	B000084EKL	A3INRM1QVW21W9	Jersey Mom	1	1	positive	1318723200	pr

In [58]:

```
final_afterRemovingDuplicating=final_afterRemovingDuplicating[final_afterRemovingDuplicating["Help
fulnessNumerator"]<=final_afterRemovingDuplicating["HelpfulnessDenominator"]]
```

In [59]:

```
final_afterRemovingDuplicating.shape
```

Out[59]:

```
(7975, 10)
```

value_counts() function gived no. of positive is 6658 and no. of negative is 1317 totaling to 7975 datapoints

In [60]:

```
final_afterRemovingDuplicating["Score"].value_counts()
```

Out[60]:

```
positive    6658
negative    1317
Name: Score, dtype: int64
```

Removing the ID column as it has no importance. Hence axis=1 column wise.

In [61]:

```
final_afterRemovingDuplicating=final_afterRemovingDuplicating.drop("Id",axis=1)
```

Important Observation: # before doing the text preprocessing checking the data by picking the data randomly. # Observation: # 1.Special character like ?,$,[....],--,comma," etc. to be removed. # 2.All character to be converted to lower case. #3.anything between html tag < > to be removed.

Applying the NLTK package and BeautifulSoup package to remove html tag,special character,alphanemeric character,web address etc. as a part of preprocessing step

In [81]:

```
from nltk.corpus import words
preprocessed_reviews=[]
#from tqdm import tqdm
for sentance in final_afterRemovingDuplicating['Text'].values:
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)

    preprocessed_reviews.append(sentance)
```

Converting preprocessed reviews in lower case character.

In [84]:

```
lowercase_review = [v.lower() for v in preprocessed_reviews]
```

checking randomly how sample review text looks after applying preprocessing step.

In [85]:

```
lowercase_review[4975]
```

Out[85]:

```
'too soft i just used this sugar to bake one of the layers for a firm pound cake like layer cake i
ntended for decoration wilton butter cake recipe this is a pretty hardy cake and can stand up to c
hanges in temperature etc and still turn out well i had already baked several of the layers with r
egular granulated sugar before i ran out and decided to use some of this baking sugar be
forewarned this sugar makes cakes far too tender the layer literally fell apart coming out of the
pan and it was well cooled and in a pan liberally covered in cake release even when i was mixing t
he batter it for some reason stuck to my stainless bowl and was impossible to scrape entirely out
i m sure this is good for meringues and such but i would never attempt to use it again for a cake
or even cookies you would never get them out of or off the pan '
```

Assiging set of words as stopwords.

URL:

In [89]:

```
stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "y
ou're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his',
'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them',
'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll",
'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having',
'do', 'does', \
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', '
while', 'of', \
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during',
'before', 'after',\
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under'
, 'again', 'further',\
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'e
ach', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'very', \
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll'
, 'm', 'o', 're', \
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "d
esn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
"mightn't", 'mustn',\
```

```
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn',
"wasn't", 'weren', "weren't", \
            'won', "won't", 'wouldn', "wouldn't"])
```

Applying the function to remove the stopword from review text.

In [92]:

```
#Source code copied from:https://stackoverflow.com/questions/49701415/is-it-possible-to-add-two-wo
rds-together-while-counting-the-word-frequencies-py
final_afterRemovingDuplicating["text"]=final_afterRemovingDuplicating["text"].apply(lambda x:" ".jo
in([word for word in x.split()if word not in stopwords]))
```

Applying the bag of word(BOW) technique on review text.

In [104]:

```
#Bag of words technique.
from sklearn.feature_extraction.text import CountVectorizer
```

In [105]:

```
vectorizer = CountVectorizer()
```

In [106]:

```
final_counts=vectorizer.fit_transform(final_afterRemovingDuplicating["text"])
```

In [107]:

```
#16507 unique words are there in sample 8000 reviews.
final_counts.get_shape()
```

Out[107]:

```
(7975, 16507)
```

applying the unigram,bigram and n-gram technique

In [108]:

```
vectorizer = CountVectorizer(ngram_range=(1,2),min_df=10,max_features=5000)
```

In [109]:

```
final_bigram_counts = vectorizer.fit_transform(final_afterRemovingDuplicating["text"])
```

In [110]:

```
final_bigram_counts.get_shape()
```

Out[110]:

```
(7975, 4755)
```

In [111]:

```
type(final_bigram_counts)
```

Out[111]:

```
scipy.sparse.csr.csr_matrix
```

Converting the sparse matrix to dense matrix. As TSNE accepts only dense matrix.

In [118]:

```
final_bigram_counts=final_bigram_counts.todense()
```

Sklearn is package available to write the machine learning algorithm. Hence importing sklearn package and then importing TSNE module from sklearn to reduce the dimension.

In [117]:

```
from sklearn.manifold import TSNE
```

Setting parameter for TSNE model like dimensions,neighbourhood points,iteration size etc.

In [133]:

```
model_TSNE=TSNE(n_components=2,perplexity=50,learning_rate=1000,n_iter=1000,random_state=0)
```

Transforming the 50 dimentions data to 2 dimensions data using TSNE model.

In [134]:

```
tsne_transformed_data=model_TSNE.fit_transform(final_bigram_counts)
```

Arranging 2 dimensions and output_label row wise.

In [135]:

```
tsne_data_array_format=np.vstack((tsne_transformed_data.T,final_afterRemovingDuplicating["Score"])
).T
```

Loading the data in 3 column of the dataframe tse_transformed_dataframe.

In [136]:

```
tse_transformed_dataframe=pd.DataFrame(data=tsne_data_array_format,columns=("Dimension_X","Dimensic
n_Y","Review Segregation"))
```

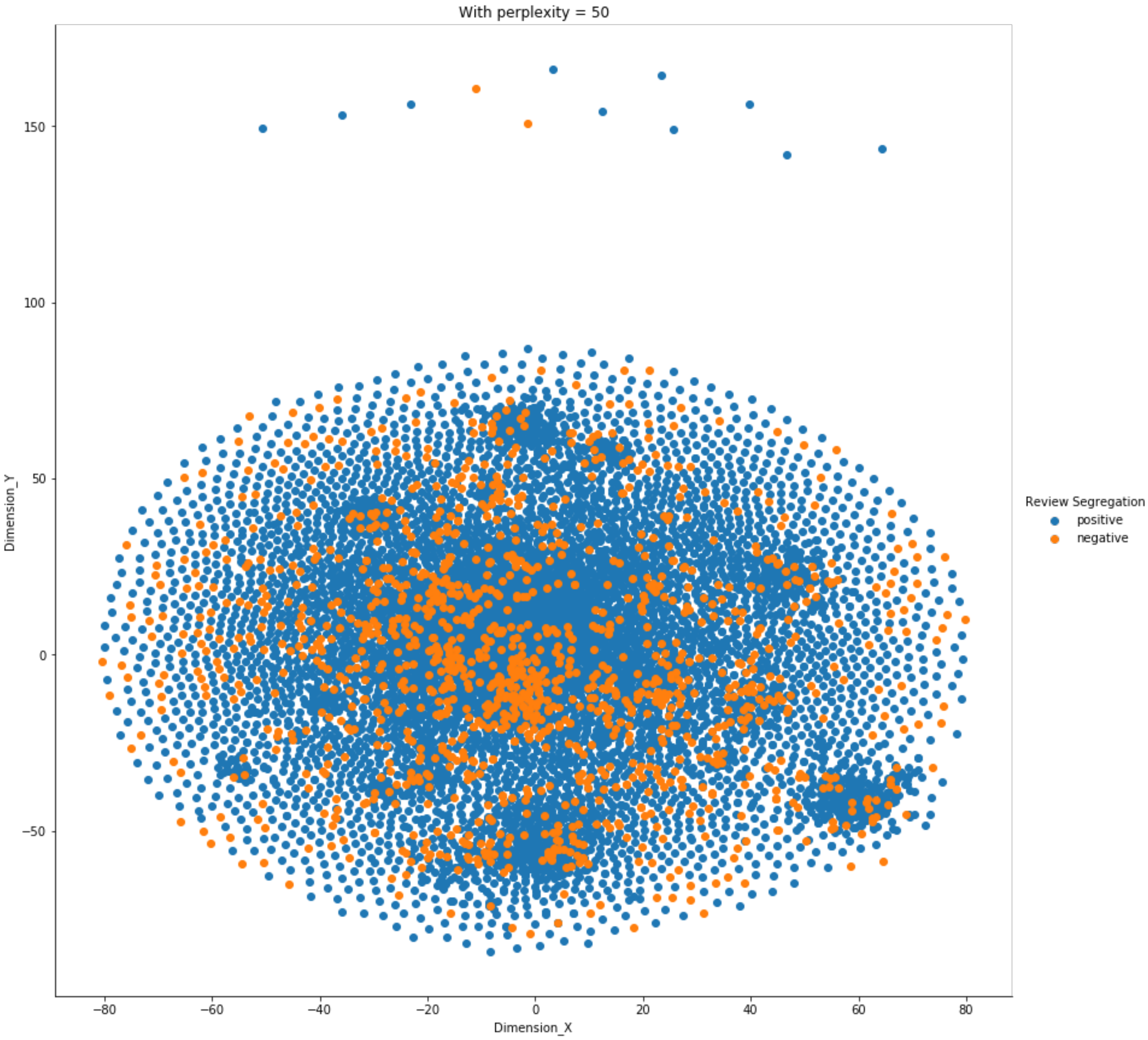
Plotting the data using Seaborn

In [137]:

```
import seaborn as sn
```

In [138]:

```
g=sn.FacetGrid(tse_transformed_dataframe,hue="Review Segregation",size=12)
g.map(plt.scatter, "Dimension_X", "Dimension_Y").add_legend()
plt.title("With perplexity = 50")
plt.show()
```



Observation 1:

Bag of words technique waht happen is it divide the sentence into words without considering the semantic meaning into account. Then it represent it to matrix by refering a word which is repeating number of times.

Cleaerly using BOW technique we cannot visualize positive and negative reviews separately. It is overlapping.

Using TF-IDF technique

tf-idf will give more focus on the word which is very rare in the whole document corpus and the same word which is more frequent in document.

In [139]:

```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2),min_df=10)
```

In [145]:

```
type(tf_idf_final_count)
```

Out[145]:

```
scipy.sparse.csr.csr_matrix
```

Converting the sparse matrix to dense matrix. As TSNE accepts only dense matrix.

In [146]:

```
tf_idf_final_count=tf_idf_final_count.todense()
```

Sklearn is package available to write the machine learning algorithm. Hence importing sklearn package and then importing TSNE module from sklearn.

In [147]:

```
from sklearn.manifold import TSNE
```

Setting parameter for TSNE model like dimensions,neighbourhood points,iteration size etc.

In [148]:

```
model_TSNE=TSNE(n_components=2,perplexity=50,learning_rate=1000,n_iter=1000,random_state=0)
```

Transforming the multiple dimentions data to 2 dimensions data using TSNE model.

In [149]:

```
tsne_transformed_data=model_TSNE.fit_transform(tf_idf_final_count)
```

Arranging 2 dimensions and output_label row wise.

In [150]:

```
tsne_data_array_format=np.vstack((tsne_transformed_data.T,final_afterRemovingDuplicating["Score"])
).T
```

Loading the data in 3 column of the dataframe tse_transformed_dataframe.

In [151]:

```
tse_transformed_dataframe=pd.DataFrame(data=tsne_data_array_format,columns=("Dimension_X","Dimensic
n_Y","Review Segregation"))
```

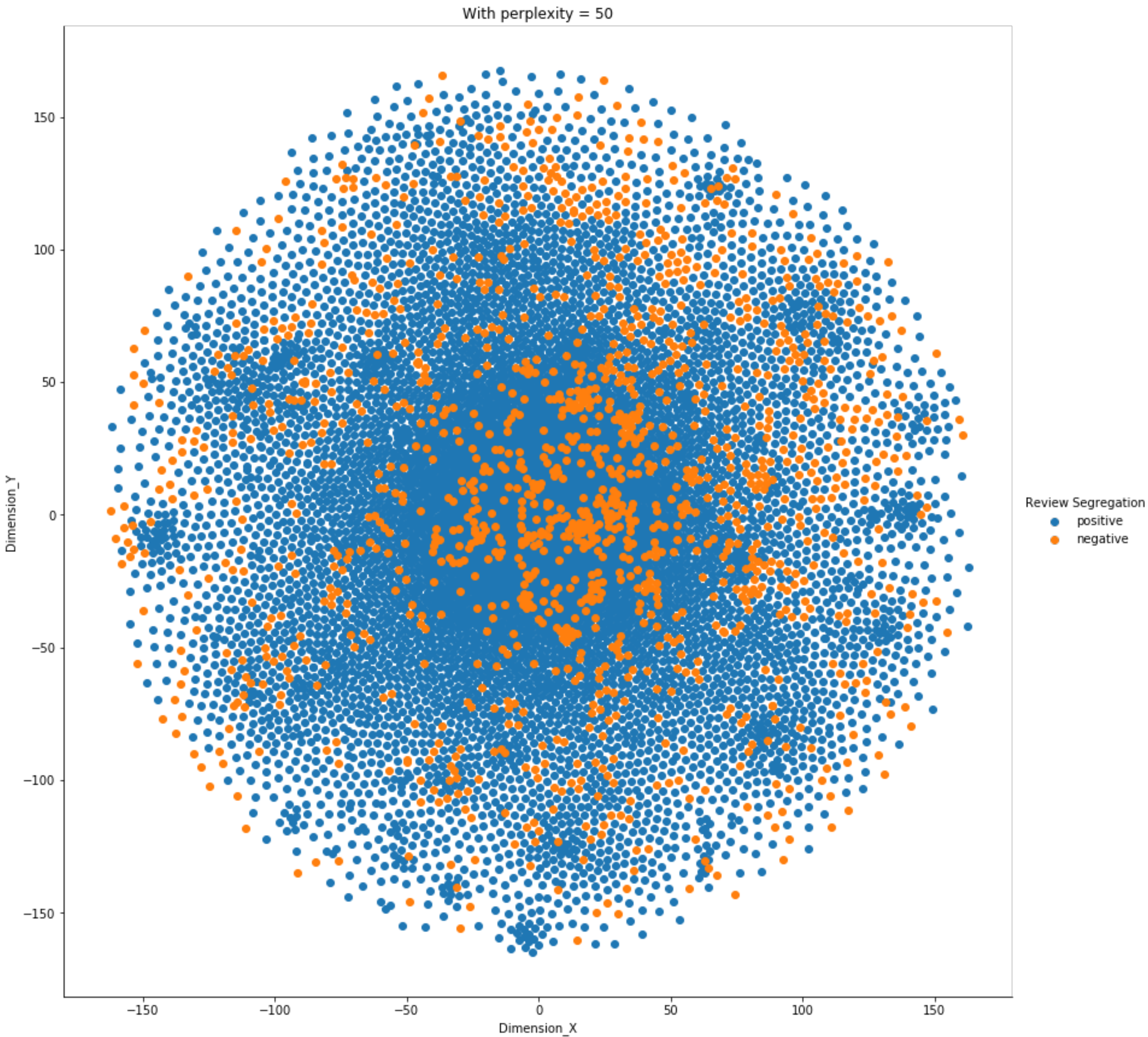
Plotting the data

In [152]:

```
import seaborn as sn
```

In [153]:

```
g=sn.FacetGrid(tse_transformed_dataframe,hue="Review Segregation",size=12)
g.map(plt.scatter, "Dimension_X", "Dimension_Y").add_legend()
plt.title("With perplexity = 50")
plt.show()
```



Observation 2:

TF_IDF will focus on word which is very rare in the whole document and the same word which is repetitive review text.
Clearly using BOW technique we cannot visualize positive and negative reviews separately. It is overlapping.

Applying the Average word2vec model on text review.

In [174]:

```
#average word2vec

# 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 tqdm(list_of_sentance): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, you might need to change this
to 300 if you use google's w2v
    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]))
```

```
100%|███████████████████████████████████████████████████████████████████████████████| 7975/7975
[00:43<00:00, 182.86it/s]
```

```
7975
50
```

Sklearn is package available to write the machine learning algorithm. Hence importing sklearn package and then importing TSNE module from sklearn.

In [176]:

```
from sklearn.manifold import TSNE
```

Setting parameter for TSNE model like dimensions,neighbourhood points,iteration size etc.

In [177]:

```
model_TSNE=TSNE(n_components=2,perplexity=50,learning_rate=1000,n_iter=1000,random_state=0)
```

Transforming the multiple dimentions data to 2 dimensions data using TSNE model.

In [179]:

```
tsne_transformed_data=model_TSNE.fit_transform(sent_vectors)
```

arranging 2 dimensions and output_label row wise.

In [187]:

```
tsne_data_array_format=np.vstack((tsne_transformed_data.T,final_afterRemovingDuplicating["Score"])
).T
```

Loading the data in 3 column of the dataframe tse_transformed_dataframe.

In [188]:

```
tse_transformed_dataframe=pd.DataFrame(data=tsne_data_array_format,columns=("Dimension_X","Dimensio
n_Y","Review_Segregation"))
```

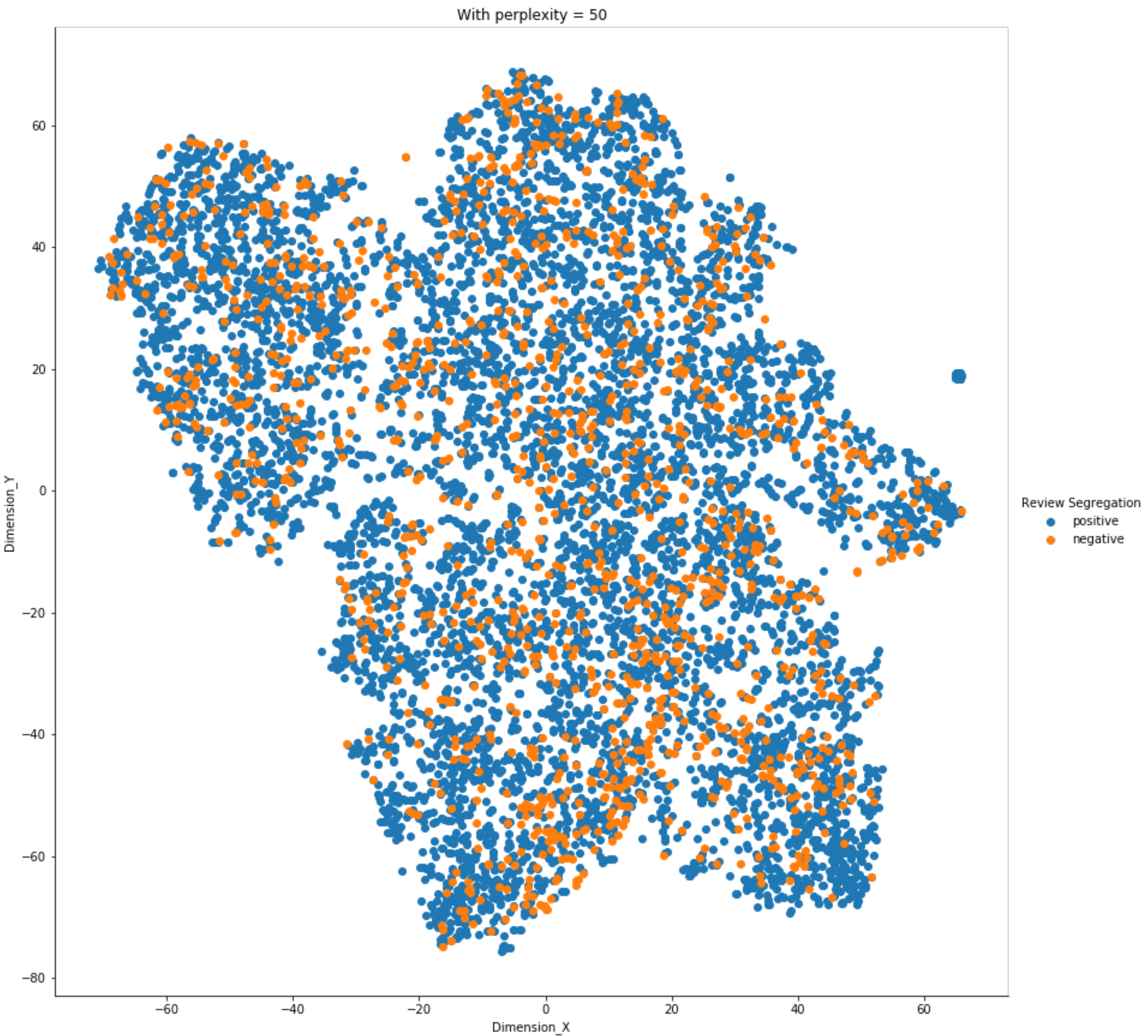
Plotting the data

In [189]:

```
import seaborn as sn
```

In [183]:

```
g=sn.FacetGrid(tse_transformed_dataframe,hue="Review Segregation",size=12)
g.map(plt.scatter, "Dimension_X", "Dimension_Y").add_legend()
plt.title("With perplexity = 50")
plt.show()
```



Observation 3:

This Word2vec model will take semantic meaning into cosideration while representing text in to vectors.
Clearly using BOW technique we cannot visualize positive and negative reviews separately. It is overlapping.

Applying th TF-IDFWeighted Word2vec model.

In [190]:

```
model = TfidfVectorizer()
model.fit(final_afterRemovingDuplicating["text"])
# 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 [191]:

```
# TF-IDF weighted Word2Vec
tfidf_feat = model.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 list
row=0;
for sent in tqdm(list_of_sentance): # 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 and word in tfidf_feat:
            vec = w2v_model.wv[word]
#             tf_idf = tf_idf_matrix[row, tfidf_feat.index(word)]
            # to reduce the computation we are
            # dictionary[word] = idf value of word in whole courpus
            # sent.count(word) = tf valeus of word in this review
            tf_idf = dictionary[word]*(sent.count(word)/len(sent))
            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
```

```
100%|████████████████████████████████████████████████████████████████████████████████| 7975/7975 [06
:24<00:00, 20.72it/s]
```

Sklearn is package available to write the machine learning algorithm. Hence importing sklearn package and then importing TSNE module from sklearn.

In [192]:

```
from sklearn.manifold import TSNE
```

Setting parameter for TSNE model like dimensions,neighbourhood points,iteration size etc.

In [193]:

```
model_TSNE=TSNE(n_components=2,perplexity=50,learning_rate=1000,n_iter=1000,random_state=0)
```

Transforming the multiple dimentions data to 2 dimensions data using TSNE model.

In [194]:

```
tsne_transformed_data=model_TSNE.fit_transform(tfidf_sent_vectors)
```

arranging 2 dimensions and output_label row wise.

In [195]:

```
tsne_data_array_format=np.vstack((tsne_transformed_data.T,final_afterRemovingDuplicating["Score"])
).T
```

Loading the data in 3 column of the dataframe tse_transformed_dataframe.

In [196]:

```
tse_transformed_dataframe=pd.DataFrame(data=tsne_data_array_format,columns=("Dimension_X","Dimensic
n_Y","Review Segregation"))
```

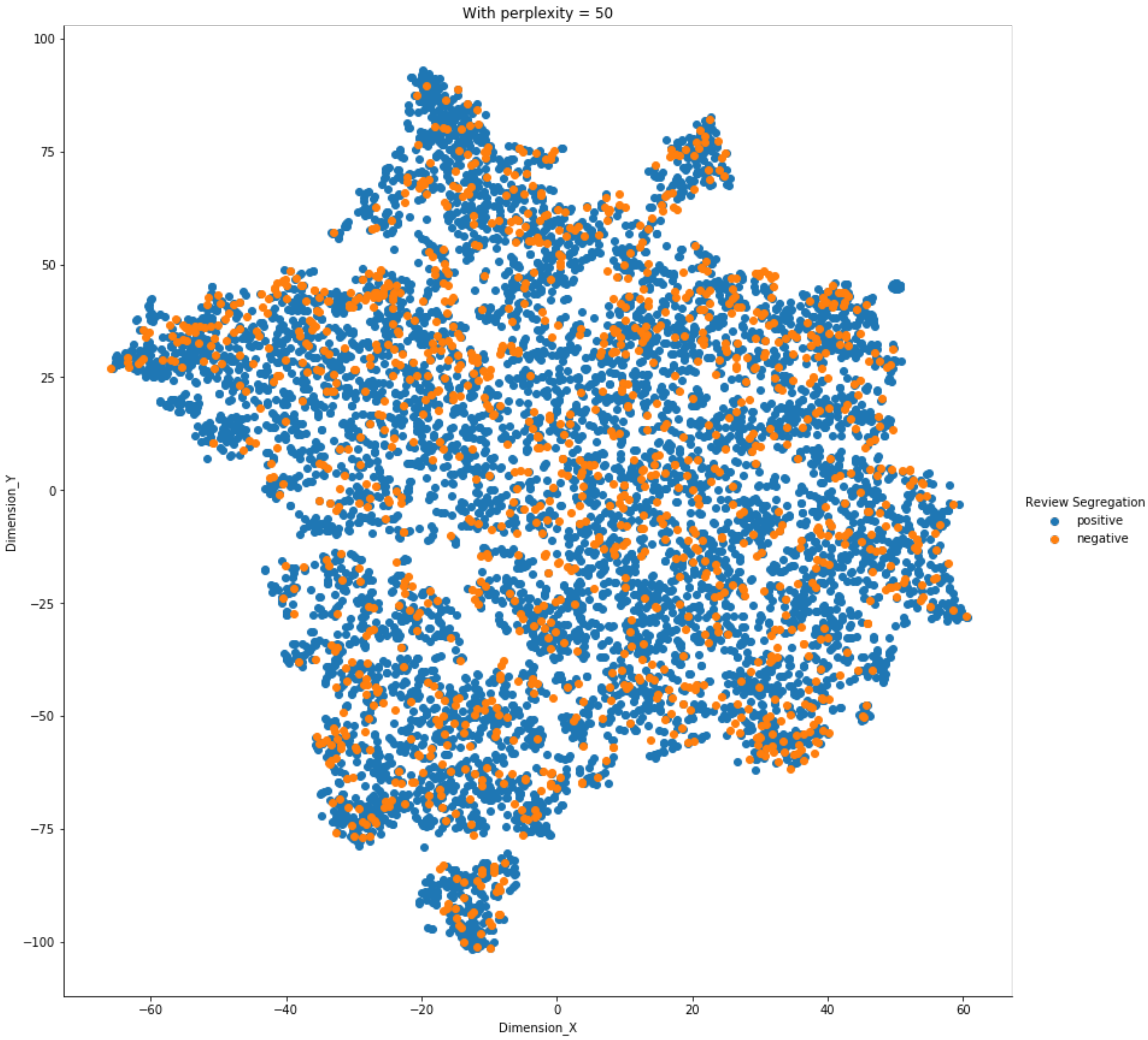
Plotting the data using Seaborn.

In [197]:

```
import seaborn as sn
```

In [198]:

```
g=sn.FacetGrid(tse_transformed_dataframe,hue="Review Segregation",size=12)
g.map(plt.scatter, "Dimension_X", "Dimension_Y").add_legend()
plt.title("With perplexity = 50")
plt.show()
```



Conclusion-

Clearly data is overlapping. However by seeing the plot we can interpret Word2vec and is better than BOW and TFIDF.