Amazon Fine Food Reviews Analysis

Data Source: https://www.kaggle.com/snap/amazon-fine-food-reviews

EDA: https://nycdatascience.com/blog/student-works/amazon-fine-foods-visualization/

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 - unqiue identifier for the user
4. ProfileName
5. HelpfulnessNumerator - number of users who found the review helpful
6. HelpfulnessDenominator - number of users who indicated whether they found the review helpful or not
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).

[Q] How to determine if a review is positive or negative?

[Ans] We could use Score/Rating. A rating of 4 or 5 can be cosnidered as a positive review. A rating of 1 or 2 can be considered as negative one. A review of rating 3 is considered nuetral and such reviews are ignored from our analysis. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

[1]. Reading Data

[1.1] Loading the data

The dataset is available in two forms

1. .csv file
2. SQLite Database

In order to load the data, We have used the SQLITE dataset as it is easier to query the data and visualise the data efficiently.

Here as we only want to get the global sentiment of the recommendations (positive or negative), we will purposefully ignore all Scores equal to 3. If the score is above 3, then the recommendation wil be set to "positive". Otherwise, it will be set to "negative".

In [29]:
```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")


import sqlite3
import pandas as pd
import numpy as np
import nltk
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, roc_auc_score
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
from collections import Counter
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_validate
```

In [30]:

```
# using SQLite Table to read data.
con = sqlite3.connect('/Users/puravshah/Downloads/amazon-fine-food-reviews/database.sqlite')
```

```
# filtering only positive and negative reviews i.e.
# not taking into consideration those reviews with Score=3
# SELECT * FROM Reviews WHERE Score != 3 LIMIT 500000, will give top 50
0000 data points
# you can change the number to any other number based on your computing
 power

# filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Sco
re != 3 LIMIT 500000""", con)
# for tsne assignment you can take 5k data points

filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score
 != 3 LIMIT 5000""", con)

# Give reviews with Score>3 a positive rating(1), and reviews with a sc
ore<3 a negative rating(0).
def partition(x):
    if x < 3:
        return 0
    return 1

#changing reviews with score less than 3 to be positive and vice-versa
actualScore = filtered_data['Score']
positiveNegative = actualScore.map(partition)
filtered_data['Score'] = positiveNegative
print("Number of data points in our data", filtered_data.shape)
filtered_data.head(3)
```

```
Number of data points in our data (5000, 10)
```

Out[30]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulnes
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfulnes
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1

In [31]:
```
display = pd.read_sql_query("""
SELECT UserId, ProductId, ProfileName, Time, Score, Text, COUNT(*)
FROM Reviews
GROUP BY UserId
HAVING COUNT(*)>1
""", con)
```

In [32]:
```
print(display.shape)
display.head()
```

```
(80668, 7)
```

Out[32]:

	UserId	ProductId	ProfileName	Time	Score	Text	COU

	UserId	ProductId	ProfileName	Time	Score	Text	COUI
0	#oc-R115TNMSPFT9I7	B007Y59HVM	Breyton	1331510400	2	Overall its just OK when considering the price...	2
1	#oc-R11D9D7SHXIJB9	B005HG9ET0	Louis E. Emory "hoppy"	1342396800	5	My wife has recurring extreme muscle spasms, u...	3
2	#oc-R11DNU2NBKQ23Z	B007Y59HVM	Kim Cieszykowski	1348531200	1	This coffee is horrible and unfortunately not ...	2
3	#oc-R11O5J5ZVQE25C	B005HG9ET0	Penguin Chick	1346889600	5	This will be the bottle that you grab from the...	3
4	#oc-R12KPBODL2B5ZD	B007OSBE1U	Christopher P. Presta	1348617600	1	I didnt like this coffee. Instead of telling y...	2

In [33]:
```
display[display['UserId']=='AZY10LLTJ71NX']
```

Out[33]:

	UserId	ProductId	ProfileName	Time	Score	Text	C

	UserId	ProductId	ProfileName	Time	Score	Text
80638	AZY10LLTJ71NX	B006P7E5ZI	undertheshrine "undertheshrine"	1334707200	5	I was recommended to try green tea extract to ...

In [34]:
```
display['COUNT(*)'].sum()
```

Out[34]:
```
393063
```

[2] Exploratory Data Analysis

[2.1] Data Cleaning: Deduplication

It is observed (as shown in the table below) that the reviews data had many duplicate entries. Hence it was necessary to remove duplicates in order to get unbiased results for the analysis of the data. Following is an example:

In [35]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[35]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	Helpfuln

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

As it can be seen above that same user has multiple reviews with same values for HelpfulnessNumerator, HelpfulnessDenominator, Score, Time, Summary and Text and on doing analysis it was found that

ProductId=B000HDOPZG was Loacker Quadratini Vanilla Wafer Cookies, 8.82-Ounce Packages (Pack of 8)

ProductId=B000HDL1RQ was Loacker Quadratini Lemon Wafer Cookies, 8.82-Ounce Packages (Pack of 8) and so on

It was inferred after analysis that reviews with same parameters other than ProductId belonged to the same product just having different flavour or quantity. Hence in order to reduce redundancy it was decided to eliminate the rows having same parameters.

The method used for the same was that we first sort the data according to ProductId and then just keep the first similar product review and delelte the others. for eg. in the above just the review for ProductId=B000HDL1RQ remains. This method ensures that there is only one representative for each product and deduplication without sorting would lead to possibility of different representatives still existing for the same product.

In [36]:
```
#Sorting data according to ProductId in ascending order
sorted_data=filtered_data.sort_values('ProductId', axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last')
```

In [37]:
```
#Deduplication of entries
final=sorted_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, keep='first', inplace=False)
final.shape
```

Out[37]:
```
(4986, 10)
```

In [38]:
```
#Checking to see how much % of data still remains
(final['Id'].size*1.0)/(filtered_data['Id'].size*1.0)*100
```

Out[38]:
```
99.72
```

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 calcualtions

In [39]:
```
display= pd.read_sql_query("""
SELECT *
FROM Reviews
WHERE Score != 3 AND Id=44737 OR Id=64422
ORDER BY ProductID
""", con)

display.head()
```

Out[39]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulIn
0	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1
1	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2

In [40]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

In [41]:
```
#Before starting the next phase of preprocessing lets see the number of
 entries left
print(final.shape)
```

```
#How many positive and negative reviews are present in our dataset?
final['Score'].value_counts()
```

```
(4986, 10)
```

Out[41]:
```
1    4178
0     808
Name: Score, dtype: int64
```

[3] Preprocessing

[3.1]. Preprocessing Review Text

Now that we have finished deduplication our data requires some preprocessing before we go on further with analysis and making the prediction model.

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

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

After which we collect the words used to describe positive and negative reviews

In [42]:
```
# printing some random reviews
sent_0 = final['Text'].values[0]
print(sent_0)
print("="*50)
```

```
sent_1000 = final['Text'].values[1000]
print(sent_1000)
print("="*50)

sent_1500 = final['Text'].values[1500]
print(sent_1500)
print("="*50)

sent_4900 = final['Text'].values[4900]
print(sent_4900)
print("="*50)
```

```
Why is this $[...] when the same product is available for $[...] here?<
br />http://www.amazon.com/VICTOR-FLY-MAGNET-BAIT-REFILL/dp/B00004RBDY<
br /><br />The Victor M380 and M502 traps are unreal, of course -- tota
l fly genocide. Pretty stinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious t
hese chips are.  The best thing was that there were a lot of "brown" ch
ips in the bsg (my favorite), so I bought some more through amazon and
shared with family and friends.  I am a little disappointed that there
are not, so far, very many brown chips in these bags, but the flavor is
still very good.  I like them better than the yogurt and green onion fl
avor because they do not seem to be as salty, and the onion flavor is b
etter.  If you haven't eaten Kettle chips before, I recommend that you
try a bag before buying bulk.  They are thicker and crunchier than Lays
but just as fresh out of the bag.
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what the
y were ordering; the other wants crispy cookies.  Hey, I'm sorry; but t
hese reviews do nobody any good beyond reminding us to look  before ord
ering.<br /><br />These are chocolate-oatmeal cookies.  If you don't li
ke that combination, don't order this type of cookie.  I find the combo
quite nice, really.  The oatmeal sort of "calms" the rich chocolate fla
vor and gives the cookie sort of a coconut-type consistency.  Now let's
also remember that tastes differ; so, I've given my opinion.<br /><br /
>Then, these are soft, chewy cookies -- as advertised.  They are not "c
rispy" cookies, or the blurb would say "crispy," rather than "chewy."
I happen to like raw cookie dough; however, I don't see where these tas
```

```
te like raw cookie dough.  Both are soft, however, so is this the confu
sion?  And, yes, they stick together.  Soft cookies tend to do that.  T
hey aren't individually wrapped, which would add to the cost.  Oh yeah,
chocolate chip cookies tend to be somewhat sweet.<br /><br />So, if you
want something hard and crisp, I suggest Nabiso's Ginger Snaps.  If you
want a cookie that's soft, chewy and tastes like a combination of choco
late and oatmeal, give these a try.  I'm here to place my second order.
==================================================
love to order my coffee on amazon.  easy and shows up quickly.<br />Thi
s k cup is great coffee.  dcaf is very good as well
==================================================
```

In [43]:
```
# remove urls from text python: https://stackoverflow.com/a/40823105/40
84039
sent_0 = re.sub(r"http\S+", "", sent_0)
sent_1000 = re.sub(r"http\S+", "", sent_1000)
sent_150 = re.sub(r"http\S+", "", sent_1500)
sent_4900 = re.sub(r"http\S+", "", sent_4900)

print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<
br /> /><br />The Victor M380 and M502 traps are unreal, of course -- t
otal fly genocide. Pretty stinky, but only right nearby.
```

In [44]:
```
# https://stackoverflow.com/questions/16206380/python-beautifulsoup-how
-to-remove-all-tags-from-an-element
from bs4 import BeautifulSoup

soup = BeautifulSoup(sent_0, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_1000, 'lxml')
text = soup.get_text()
print(text)
print("="*50)
```

```
soup = BeautifulSoup(sent_1500, 'lxml')
text = soup.get_text()
print(text)
print("="*50)

soup = BeautifulSoup(sent_4900, 'lxml')
text = soup.get_text()
print(text)
```

```
Why is this $[...] when the same product is available for $[...] here?
/>The Victor M380 and M502 traps are unreal, of course -- total fly gen
ocide. Pretty stinky, but only right nearby.
==================================================
I recently tried this flavor/brand and was surprised at how delicious t
hese chips are.  The best thing was that there were a lot of "brown" ch
ips in the bsg (my favorite), so I bought some more through amazon and
shared with family and friends.  I am a little disappointed that there
are not, so far, very many brown chips in these bags, but the flavor is
still very good.  I like them better than the yogurt and green onion fl
avor because they do not seem to be as salty, and the onion flavor is b
etter.  If you haven't eaten Kettle chips before, I recommend that you
try a bag before buying bulk.  They are thicker and crunchier than Lays
but just as fresh out of the bag.
==================================================
Wow.  So far, two two-star reviews.  One obviously had no idea what the
y were ordering; the other wants crispy cookies.  Hey, I'm sorry; but t
hese reviews do nobody any good beyond reminding us to look  before ord
ering.These are chocolate-oatmeal cookies.  If you don't like that comb
ination, don't order this type of cookie.  I find the combo quite nice,
really.  The oatmeal sort of "calms" the rich chocolate flavor and give
s the cookie sort of a coconut-type consistency.  Now let's also rememb
er that tastes differ; so, I've given my opinion.Then, these are soft,
chewy cookies -- as advertised.  They are not "crispy" cookies, or the
blurb would say "crispy," rather than "chewy."  I happen to like raw co
okie dough; however, I don't see where these taste like raw cookie doug
h.  Both are soft, however, so is this the confusion?  And, yes, they s
tick together.  Soft cookies tend to do that.  They aren't individually
wrapped, which would add to the cost.  Oh yeah, chocolate chip cookies
tend to be somewhat sweet.So, if you want something hard and crisp, I s
uggest Nabiso's Ginger Snaps.  If you want a cookie that's soft, chewy
```

```
and tastes like a combination of chocolate and oatmeal, give these a tr
y.  I'm here to place my second order.
==================================================
love to order my coffee on amazon.  easy and shows up quickly.This k cu
p is great coffee.  dcaf is very good as well
```

In [45]:
```python
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [46]:
```python
sent_1500 = decontracted(sent_1500)
print(sent_1500)
print("="*50)
```

```
Wow.  So far, two two-star reviews.  One obviously had no idea what the
y were ordering; the other wants crispy cookies.  Hey, I am sorry; but
these reviews do nobody any good beyond reminding us to look  before or
dering.<br /><br />These are chocolate-oatmeal cookies.  If you do not
like that combination, do not order this type of cookie.  I find the co
mbo quite nice, really.  The oatmeal sort of "calms" the rich chocolate
flavor and gives the cookie sort of a coconut-type consistency.  Now le
t is also remember that tastes differ; so, I have given my opinion.<br
/><br />Then, these are soft, chewy cookies -- as advertised.  They are
not "crispy" cookies, or the blurb would say "crispy," rather than "che
```

```
wy."  I happen to like raw cookie dough; however, I do not see where th
ese taste like raw cookie dough.  Both are soft, however, so is this th
e confusion?  And, yes, they stick together.  Soft cookies tend to do t
hat.  They are not individually wrapped, which would add to the cost.
Oh yeah, chocolate chip cookies tend to be somewhat sweet.<br /><br />S
o, if you want something hard and crisp, I suggest Nabiso is Ginger Sna
ps.  If you want a cookie that is soft, chewy and tastes like a combina
tion of chocolate and oatmeal, give these a try.  I am here to place my
second order.
==================================================
```

In [47]:
```
#remove words with numbers python: https://stackoverflow.com/a/1808237
0/4084039
sent_0 = re.sub("\S*\d\S*", "", sent_0).strip()
print(sent_0)
```

```
Why is this $[...] when the same product is available for $[...] here?<
br /> /><br />The Victor  and  traps are unreal, of course -- total fly
genocide. Pretty stinky, but only right nearby.
```

In [48]:
```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent_1500 = re.sub('[^A-Za-z0-9]+', ' ', sent_1500)
print(sent_1500)
```

```
Wow So far two two star reviews One obviously had no idea what they wer
e ordering the other wants crispy cookies Hey I am sorry but these revi
ews do nobody any good beyond reminding us to look before ordering br b
r These are chocolate oatmeal cookies If you do not like that combinati
on do not order this type of cookie I find the combo quite nice really
The oatmeal sort of calms the rich chocolate flavor and gives the cooki
e sort of a coconut type consistency Now let is also remember that tast
es differ so I have given my opinion br br Then these are soft chewy co
okies as advertised They are not crispy cookies or the blurb would say
crispy rather than chewy I happen to like raw cookie dough however I do
not see where these taste like raw cookie dough Both are soft however s
o is this the confusion And yes they stick together Soft cookies tend t
o do that They are not individually wrapped which would add to the cost
Oh yeah chocolate chip cookies tend to be somewhat sweet br br So if yo
u want something hard and crisp I suggest Nabiso is Ginger Snaps If you
```

```
want a cookie that is soft chewy and tastes like a combination of choco
late and oatmeal give these a try I am here to place my second order
```

In [49]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'no
t'
# <br /><br /> ==> after the above steps, we are getting "br br"
# we are including them into stop words list
# instead of <br /> if we have <br/> these tags would have revmoved in
 the 1st step

stopwords= set(['br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'o
urs', 'ourselves', 'you', "you're", "you've",\
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselve
s', 'he', 'him', 'his', 'himself', \
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'it
s', 'itself', 'they', 'them', 'their',\
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'th
is', 'that', "that'll", 'these', 'those', \
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'h
ave', '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', 'h
ow', 'all', 'any', 'both', 'each', 'few', 'more',\
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 's
o', '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', "doesn't", 'hadn',\
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "is
n'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"])
```

In [50]:
```
# Combining all the above stundents
from tqdm import tqdm
preprocessed_reviews = []
# tqdm is for printing the status bar
for sentance in tqdm(final['Text'].values):
    sentance = re.sub(r"http\S+", "", sentance)
    sentance = BeautifulSoup(sentance, 'lxml').get_text()
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)
    # https://gist.github.com/sebleier/554280
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower
() not in stopwords)
    preprocessed_reviews.append(sentance.strip())
```

```
100%|██████████| 4986/4986 [00:01<00:00, 3420.65it/s]
```

In [51]:
```
preprocessed_reviews[1500]
```

Out[51]:
```
'wow far two two star reviews one obviously no idea ordering wants cris
py cookies hey sorry reviews nobody good beyond reminding us look order
ing chocolate oatmeal cookies not like combination not order type cooki
e find combo quite nice really oatmeal sort calms rich chocolate flavor
gives cookie sort coconut type consistency let also remember tastes dif
fer given opinion soft chewy cookies advertised not crispy cookies blur
b would say crispy rather chewy happen like raw cookie dough however no
t see taste like raw cookie dough soft however confusion yes stick toge
ther soft cookies tend not individually wrapped would add cost oh yeah
chocolate chip cookies tend somewhat sweet want something hard crisp su
ggest nabiso ginger snaps want cookie soft chewy tastes like combinatio
n chocolate oatmeal give try place second order'
```

In [114]:
```
final['Preprocessed_reviews']=preprocessed_reviews
final.drop('Text',axis=1)
print(final.head())
```

```
        Id   ProductId          UserId                      ProfileName  \
2546  2774  B00002NCJC  A196AJHU9EASJN                    Alex Chaffee
2547  2775  B00002NCJC  A13RRPGE79XFFH                        reader48
1145  1244  B00002Z754  A3B8RCEI0FXFI6                      B G Chase
1146  1245  B00002Z754  A29Z5PI9BW2PU3                        Robbie
2942  3204  B000084DVR  A1UGDJP1ZJWVPF  T. Moore "thoughtful reader"

      HelpfulnessNumerator  HelpfulnessDenominator  Score        Time
\
2546                     0                       0      1  1282953600

2547                     0                       0      1  1281052800

1145                    10                      10      1   962236800

1146                     7                       7      1   961718400

2942                     1                       1      1  1177977600


                           Summary  \
2546                  thirty bucks?
2547                   Flies Begone
1145  WOW Make your own 'slickers' !
1146                  Great Product
2942                    Good stuff!

                                                  Text  \
2546  Why is this $[...] when the same product is av...
2547  We have used the Victor fly bait for 3 seasons...
1145  I just received my shipment and could hardly w...
1146  This was a really good idea and the final prod...
2942  I'm glad my 45lb cocker/standard poodle puppy ...

                                Preprocessed_reviews
2546  product available victor traps unreal course t...
2547  used victor fly bait seasons ca not beat great...
1145  received shipment could hardly wait try produc...
```

```
1146  really good idea final product outstanding use...
2942  glad cocker standard poodle puppy loves stuff ...
```

[3.2] Preprocessing Review Summary

In [52]:
```
## Similartly you can do preprocessing for review summary also.
```

[4] Featurization

[4.1] BAG OF WORDS

In [53]:
```
#BoW
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler(with_mean=False)
X_1, X_test, y_1, y_test = train_test_split(preprocessed_reviews, final
['Score'], test_size=0.3, random_state=0)
X_tr, X_cv, y_tr, y_cv = train_test_split(X_1, y_1, test_size=0.3)
count_vect = CountVectorizer() #in scikit-learn
count_vect.fit(X_tr)
print("some feature names ", count_vect.get_feature_names()[:10])
print('='*50)

final_counts = count_vect.transform(X_tr)
#final_counts=scaler.fit_transform(final_counts)
X_cv_bow=count_vect.transform(X_cv)
#X_cv_bow=scaler.transform(X_cv_bow)
X_test_bow=count_vect.transform(X_test)
#X_test_bow=scaler.fit_transform(X_test_bow)
print("the type of count vectorizer ",type(final_counts))
print("the shape of out text BOW vectorizer ",final_counts.get_shape())
print("the number of unique words ", final_counts.get_shape()[1])
print(y_test.shape)
```

```
some feature names  ['aback', 'abandon', 'abby', 'abdominal', 'abilit
```

```
y', 'able', 'aboulutely', 'absence', 'absent', 'absolute']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (2443, 9209)
the number of unique words  9209
(1496,)
```

[4.2] Bi-Grams and n-Grams.

In [54]:
```
#bi-gram, tri-gram and n-gram

#removing stop words like "not" should be avoided before building n-gra
ms
# count_vect = CountVectorizer(ngram_range=(1,2))
# please do read the CountVectorizer documentation http://scikit-learn.
org/stable/modules/generated/sklearn.feature_extraction.text.CountVecto
rizer.html

# you can choose these numebrs min_df=10, max_features=5000, of your ch
oice
count_vect_gram = CountVectorizer(ngram_range=(1,2), min_df=10, max_fea
tures=5000)
final_bigram_counts = count_vect_gram.fit_transform(X_tr)
#final_bigram_counts=scaler.fit_transform(final_bigram_counts)
X_cv_ngram = count_vect_gram.transform(X_cv)
#X_cv_ngram=scaler.fit_transform(X_cv_ngram)
X_test_ngram=count_vect_gram.transform(X_test)
#X_test_ngram=scaler.fit_transform(X_test_ngram)
print("the type of count vectorizer ",type(final_bigram_counts))
print("the shape of out text BOW vectorizer ",final_bigram_counts.get_s
hape())
print("the number of unique words including both unigrams and bigrams "
, final_bigram_counts.get_shape()[1])
```

```
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text BOW vectorizer  (2443, 1653)
the number of unique words including both unigrams and bigrams  1653
```

[4.3] TF-IDF

In [55]:
```
tf_idf_vect = TfidfVectorizer(ngram_range=(1,2), min_df=10)
tf_idf_vect.fit(X_tr)

print("some sample features(unique words in the corpus)",tf_idf_vect.ge
t_feature_names()[0:10])
print('='*50)

final_tf_idf = tf_idf_vect.transform(X_tr)
#final_tf_idf=scaler.fit_transform(final_tf_idf)
X_cv_tfidf=tf_idf_vect.transform(X_cv)
#X_cv_tfidf=scaler.fit_transform(X_cv_tfidf)
X_test_tfidf=tf_idf_vect.transform(X_test)
#X_test_tfidf=scaler.fit_transform(X_test_tfidf)
print("the type of count vectorizer ",type(final_tf_idf))
print("the shape of out text TFIDF vectorizer ",final_tf_idf.get_shape
())
print("the number of unique words including both unigrams and bigrams "
, final_tf_idf.get_shape()[1])
```

```
some sample features(unique words in the corpus) ['able', 'able find',
'absolute', 'absolutely', 'according', 'acid', 'across', 'actual', 'act
ually', 'add']
==================================================
the type of count vectorizer  <class 'scipy.sparse.csr.csr_matrix'>
the shape of out text TFIDF vectorizer  (2443, 1653)
the number of unique words including both unigrams and bigrams  1653
```

[4.4] Word2Vec

In [56]:
```
# Train your own Word2Vec model using your own text corpus
i=0
list_of_sentance_train=[]
for sentance in X_tr:
    list_of_sentance_train.append(sentance.split())
```

In [57]:
```
i=0
list_of_sentance_cv=[]
for sentance in X_cv:
    list_of_sentance_cv.append(sentance.split())
```

In [58]:
```
i=0
list_of_sentance_test=[]
for sentance in X_test:
    list_of_sentance_test.append(sentance.split())
```

In [59]:
```
# Using Google News Word2Vectors

# in this project we are using a pretrained model by google
# its 3.3G file, once you load this into your memory
# it occupies ~9Gb, so please do this step only if you have >12G of ram
# we will provide a pickle file wich contains a dict ,
# and it contains all our courpus words as keys and  model[word] as val
ues
# To use this code-snippet, download "GoogleNews-vectors-negative300.bi
n"
# from https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edi
t
# it's 1.9GB in size.


# http://kavita-ganesan.com/gensim-word2vec-tutorial-starter-code/#.W17
SRFAzZPY
# you can comment this whole cell
# or change these varible according to your need


is_your_ram_gt_16g=False
want_to_use_google_w2v = False
want_to_train_w2v = True

if want_to_train_w2v:
    # min_count = 5 considers only words that occured atleast 5 times
```

```
    w2v_model_train=Word2Vec(list_of_sentance_train,min_count=5,size=50
, workers=4)
    print(w2v_model_train.wv.most_similar('great'))
    print('='*50)
    print(w2v_model_train.wv.most_similar('worst'))

elif want_to_use_google_w2v and is_your_ram_gt_16g:
    if os.path.isfile('GoogleNews-vectors-negative300.bin'):
        w2v_model=KeyedVectors.load_word2vec_format('GoogleNews-vectors
-negative300.bin', binary=True)
        print(w2v_model.wv.most_similar('great'))
        print(w2v_model.wv.most_similar('worst'))
    else:
        print("you don't have gogole's word2vec file, keep want_to_trai
n_w2v = True, to train your own w2v ")
```

```
[('sure', 0.9998070001602173), ('things', 0.9998016953468323), ('take',
0.9997977614402771), ('say', 0.9997960925102234), ('know', 0.9997959733
009338), ('cut', 0.9997949600219727), ('fresh', 0.999794602394104), ('e
xcellent', 0.9997944235801697), ('tasty', 0.9997878074645996), ('item',
0.9997873306274414)]
==================================================
[('choice', 0.9993889927864075), ('goes', 0.9993652105331421), ('gave',
0.9993583559989929), ('making', 0.9993571639060974), ('go', 0.999356031
4178467), ('baked', 0.9993386268615723), ('without', 0.999315142631530
8), ('nothing', 0.9993089437484741), ('brown', 0.9993056654930115), ('u
sing', 0.9993019104003906)]
```

In [60]:

```
w2v_words_train = list(w2v_model_train.wv.vocab)
print("number of words that occured minimum 5 times ",len(w2v_words_tra
in))
print("sample words ", w2v_words_train[0:50])
```

```
number of words that occured minimum 5 times  2542
sample words  ['fast', 'delivery', 'good', 'product', 'lemon', 'juice',
'concentrate', 'find', 'stores', 'really', 'handy', 'around', 'add', 's
auces', 'dressings', 'etc', 'pain', 'squeeze', 'fresh', 'need', 'hand',
'begin', 'first', 'package', 'came', 'damaged', 'rate', 'customer', 'se
rvice', 'replaced', 'quickly', 'delicious', 'easy', 'cook', 'ordered',
'two', 'bags', 'three', 'dogs', 'lab', 'terrier', 'arrived', 'well', 'p
```

```
ackaged', 'inside', 'cardboard', 'boxes', 'like', 'food', 'coats']
```

[4.4.1] Converting text into vectors using Avg W2V, TFIDF-W2V

[4.4.1.1] Avg W2v

In [61]:
```
sent_vectors_train = []; # the avg-w2v for each sentence/review is stor
ed in this list
for sent in tqdm(list_of_sentance_train): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u 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/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words_train:
            vec = w2v_model_train.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors_train.append(sent_vec)
print(len(sent_vectors_train))
print(len(sent_vectors_train[0]))
sent_vectors_train=scaler.fit_transform(sent_vectors_train)
```

```
100%|██████████| 2443/2443 [00:01<00:00, 2096.32it/s]
```

```
2443
50
```

In [62]:
```
sent_vectors_cv = []; # the avg-w2v for each sentence/review is stored
 in this list
for sent in tqdm(list_of_sentance_cv): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u 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/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words_train:
            vec = w2v_model_train.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors_cv.append(sent_vec)
print(len(sent_vectors_cv))
print(len(sent_vectors_cv[0]))
sent_vectors_cv=scaler.fit_transform(sent_vectors_cv)
```

```
100%|██████████| 1047/1047 [00:00<00:00, 1795.80it/s]
```

```
1047
50
```

In [63]:

```
sent_vectors_test = []; # the avg-w2v for each sentence/review is store
d in this list
for sent in tqdm(list_of_sentance_test): # for each review/sentence
    sent_vec = np.zeros(50) # as word vectors are of zero length 50, yo
u 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/re
view
    for word in sent: # for each word in a review/sentence
        if word in w2v_words_train:
            vec = w2v_model_train.wv[word]
            sent_vec += vec
            cnt_words += 1
    if cnt_words != 0:
        sent_vec /= cnt_words
    sent_vectors_test.append(sent_vec)
print(len(sent_vectors_test))
print(len(sent_vectors_test[0]))
sent_vectors_test=scaler.fit_transform(sent_vectors_test)
```

```
100%|██████████| 1496/1496 [00:00<00:00, 1927.05it/s]
```

```
1496
50
```

[4.4.1.2] TFIDF weighted W2v

In [64]:
```
# S = ["abc def pqr", "def def def abc", "pqr pqr def"]
model_train= TfidfVectorizer()
tf_idf_matrix_train = model_train.fit_transform(X_tr)
# we are converting a dictionary with word as a key, and the idf as a v
alue
dictionary_train = dict(zip(model_train.get_feature_names(), list(model
_train.idf_)))
```

In [65]:
```
# TF-IDF weighted Word2Vec
tfidf_feat_train = model_train.get_feature_names() # tfidf words/col-na
mes
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_sent_vectors_train = []; # the tfidf-w2v for each sentence/review
 is stored in this list
row=0;
for sent in tqdm(list_of_sentance_train): # for each review/sentence
    sent_vec_train = np.zeros(50) # as word vectors are of zero length
    weight_sum_train =0; # num of words with a valid vector in the sent
ence/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words_train and word in tfidf_feat_train:
            vec_train = w2v_model_train.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_train[word]*(sent.count(word)/len(sent))
            sent_vec_train += (vec * tf_idf)
            weight_sum_train += tf_idf
    if weight_sum_train != 0:
```

```
        sent_vec_train /= weight_sum_train
    tfidf_sent_vectors_train.append(sent_vec_train)
    row += 1
tfidf_sent_vectors_train =scaler.fit_transform(tfidf_sent_vectors_train
)
```

```
100%|██████████| 2443/2443 [00:06<00:00, 403.73it/s]
```

In [66]:

```
tfidf_feat_cv = model_train.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_sent_vectors_cv = []; # the tfidf-w2v for each sentence/review is
 stored in this list
row=0;
for sent in tqdm(list_of_sentance_cv): # for each review/sentence
    sent_vec_cv = np.zeros(50) # as word vectors are of zero length
    weight_sum_cv =0; # num of words with a valid vector in the sentenc
e/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words_train and word in tfidf_feat_cv:
            vec_cv = w2v_model_train.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_train[word]*(sent.count(word)/len(sent
))
            sent_vec_cv += (vec * tf_idf)
            weight_sum_cv += tf_idf
    if weight_sum_cv != 0:
        sent_vec_cv /= weight_sum_cv
    tfidf_sent_vectors_cv.append(sent_vec_cv)
    row += 1
tfidf_sent_vectors_cv =scaler.fit_transform(tfidf_sent_vectors_cv)
```

```
100%|██████████| 1047/1047 [00:02<00:00, 418.59it/s]
```

In [67]:

```
tfidf_feat_test = model_train.get_feature_names() # tfidf words/col-nam
```

```
es
# final_tf_idf is the sparse matrix with row= sentence, col=word and ce
ll_val = tfidf

tfidf_sent_vectors_test = []; # the tfidf-w2v for each sentence/review
 is stored in this list
row=0;
for sent in tqdm(list_of_sentance_test): # for each review/sentence
    sent_vec_test = np.zeros(50) # as word vectors are of zero length
    weight_sum_test =0; # num of words with a valid vector in the sente
nce/review
    for word in sent: # for each word in a review/sentence
        if word in w2v_words_train and word in tfidf_feat_test:
            vec_test = w2v_model_train.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_train[word]*(sent.count(word)/len(sent
))
            sent_vec_test += (vec * tf_idf)
            weight_sum_test += tf_idf
    if weight_sum_test != 0:
        sent_vec_test /= weight_sum_test
    tfidf_sent_vectors_test.append(sent_vec_test)
    row += 1
tfidf_sent_vectors_test =scaler.fit_transform(tfidf_sent_vectors_test)
```

```
100%|██████████| 1496/1496 [00:03<00:00, 382.90it/s]
```

[5] Assignment 10: K-Means, Agglomerative & DBSCAN Clustering

1. **Apply K-means Clustering on these feature sets:**

   - SET 1:Review text, preprocessed one converted into vectors using (BOW)
   - SET 2:Review text, preprocessed one converted into vectors using (TFIDF)

- SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
- SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)
- Find the best 'k' using the elbow-knee method (plot k vs inertia_)
- Once after you find the k clusters, plot the word cloud per each cluster so that at a single go we can analyze the words in a cluster.

2. **Apply Agglomerative Clustering on these feature sets:**

- SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
- SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)
- Apply agglomerative algorithm and try a different number of clusters like 2,5 etc.
- Same as that of K-means, plot word clouds for each cluster and summarize in your own words what that cluster is representing.
- You can take around 5000 reviews or so(as this is very computationally expensive one)

3. **Apply DBSCAN Clustering on these feature sets:**

- SET 3:Review text, preprocessed one converted into vectors using (AVG W2v)
- SET 4:Review text, preprocessed one converted into vectors using (TFIDF W2v)
- Find the best 'Eps' using the elbow-knee method.
- Same as before, plot word clouds for each cluster and summarize in your own words what that cluster is representing.
- You can take around 5000 reviews for this as well.

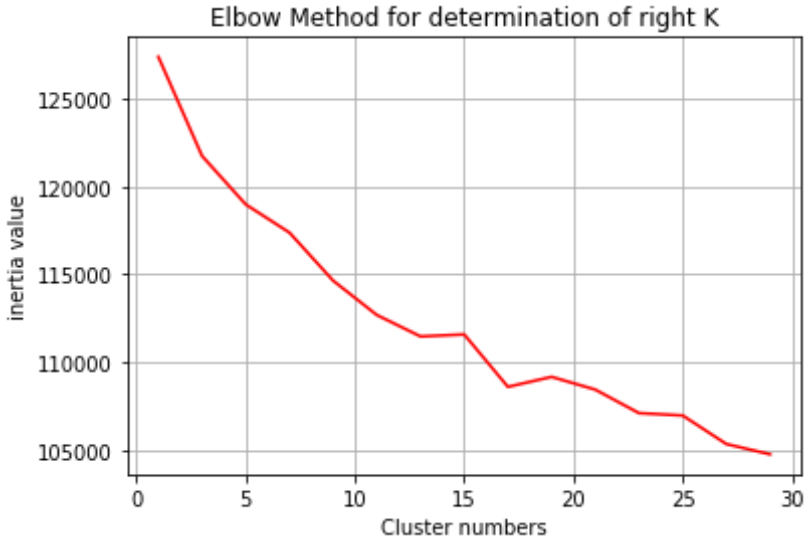
[5.1] K-Means Clustering

[5.1.1] Applying K-Means Clustering on BOW, SET 1

In [71]:
```
# Please write all the code with proper documentation
from sklearn.cluster import KMeans
k=list(range(1,30,2))
inertia=[]
for i in k:
    model=KMeans(n_clusters=i)
    model.fit(final_counts)
    model_result=model.inertia_
    inertia.append(model.inertia_)
    print('for the value of k=%d the value of inertia is %f'%(i,model_r
esult))
plt.plot(k,inertia,c='r')
plt.grid()
plt.xlabel('Cluster numbers')
plt.ylabel('inertia value')
plt.title('Elbow Method for determination of right K')
plt.show
```

```
for the value of k=1 the value of inertia is 127361.494883
for the value of k=3 the value of inertia is 121731.524474
for the value of k=5 the value of inertia is 118978.004812
for the value of k=7 the value of inertia is 117379.828337
for the value of k=9 the value of inertia is 114649.495257
for the value of k=11 the value of inertia is 112697.102379
for the value of k=13 the value of inertia is 111480.460310
for the value of k=15 the value of inertia is 111594.329594
for the value of k=17 the value of inertia is 108617.049040
for the value of k=19 the value of inertia is 109184.845228
for the value of k=21 the value of inertia is 108460.550125
for the value of k=23 the value of inertia is 107125.154225
for the value of k=25 the value of inertia is 106991.659743
for the value of k=27 the value of inertia is 105369.509090
for the value of k=29 the value of inertia is 104785.677127
```

Out[71]:
```
<function matplotlib.pyplot.show(*args, **kw)>
```



In [116]:
```
reviews=final['Preprocessed_reviews'].values
print(len(preprocessed_reviews))
model_new=KMeans(n_clusters=6)
model_new.fit(final_counts)
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif model_new.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif model_new.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif model_new.labels_[i] == 3:
        cluster4.append(reviews[i])
    elif model_new.labels_[i] == 4:
        cluster5.append(reviews[i])
```

```
    else:
        cluster6.append(reviews[i])



# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
print("\nNo. of reviews in Cluster-5 : ",len(cluster5))
print("\nNo. of reviews in Cluster-6 : ",len(cluster6))
count=1
for i in range(3):
    print('Review-%d : \n %s\n'%(count,cluster1[i]))
    count +=1
```

```
4986
No. of reviews in Cluster-1 :  12

No. of reviews in Cluster-2 :  55

No. of reviews in Cluster-3 :  496

No. of reviews in Cluster-4 :  89

No. of reviews in Cluster-5 :  1720

No. of reviews in Cluster-6 :  71
Review-1 :
 find jar product works better people without much dexterity hands vs u
sing opening sweetener packets product works well cold hot things like
homemade lemonade coffee easy open lid product dissolves easily cold li
quids many types stevia

Review-2 :
 used sons bday party one sweet treats great kids loved price fabulous
way better purchasing elsewhere item came described fresh thank amazon
com
```

```
Review-3 :
 based amazon reviews tried product first shipment great delicious thou
ght would found new healthful snack shipment concern disappointment eve
n thought calling company not several different packages incredibly har
d pieces not sure almonds became concerned would chip tooth not ordered
```

In [117]:

```
from wordcloud import WordCloud,STOPWORDS
stopwords=set(STOPWORDS)
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()
```

In [118]:
```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()
```

In [119]:
```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster3))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 3')
plt.show()
```

In [120]:
```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster4))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 4')
plt.show()
```

In [121]:
```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster5))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 5')
plt.show()
```

In [122]:
```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster6))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 6')
plt.show()
```

[5.1.2] Wordclouds of clusters obtained after applying k-means on BOW SET 1

In [3]:
```
# Please write all the code with proper documentation
```

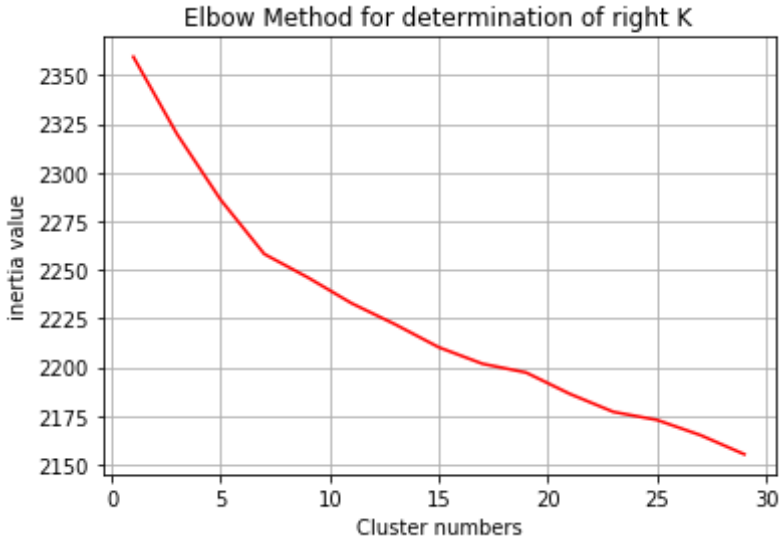
[5.1.3] Applying K-Means Clustering on TFIDF, SET 2

In [86]:
```
# Please write all the code with proper documentation
k=list(range(1,30,2))
inertia=[]
for i in k:
    model=KMeans(n_clusters=i)
    model.fit(final_tf_idf)
    model_result=model.inertia_
    inertia.append(model.inertia_)
```

```
    print('for the value of k=%d the value of inertia is %f'%(i,model_r
esult))
plt.plot(k,inertia,c='r')
plt.grid()
plt.xlabel('Cluster numbers')
plt.ylabel('inertia value')
plt.title('Elbow Method for determination of right K')
plt.show
```

```
for the value of k=1 the value of inertia is 2359.477231
for the value of k=3 the value of inertia is 2319.808270
for the value of k=5 the value of inertia is 2286.037573
for the value of k=7 the value of inertia is 2258.140637
for the value of k=9 the value of inertia is 2246.066037
for the value of k=11 the value of inertia is 2232.830713
for the value of k=13 the value of inertia is 2221.965194
for the value of k=15 the value of inertia is 2210.184255
for the value of k=17 the value of inertia is 2201.733420
for the value of k=19 the value of inertia is 2197.212836
for the value of k=21 the value of inertia is 2186.269775
for the value of k=23 the value of inertia is 2177.001004
for the value of k=25 the value of inertia is 2172.849047
for the value of k=27 the value of inertia is 2165.000792
for the value of k=29 the value of inertia is 2155.202667
```

Out[86]: <function matplotlib.pyplot.show(*args, **kw)>



In [123]:
```
reviews=final['Preprocessed_reviews'].values
model_new=KMeans(n_clusters=7)
model_new.fit(final_tf_idf)
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
cluster5 = []
cluster6 = []
cluster7 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif model_new.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif model_new.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif model_new.labels_[i] == 3:
        cluster4.append(reviews[i])
    elif model_new.labels_[i] == 4:
        cluster5.append(reviews[i])
```

```
    elif model_new.labels_[i]==5:
        cluster6.append(reviews[i])
    else:
        cluster7.append(reviews[i])




# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
print("\nNo. of reviews in Cluster-5 : ",len(cluster5))
print("\nNo. of reviews in Cluster-6 : ",len(cluster6))
print("\nNo. of reviews in Cluster-7 : ",len(cluster7))
```

```
No. of reviews in Cluster-1 :  172

No. of reviews in Cluster-2 :  821

No. of reviews in Cluster-3 :  157

No. of reviews in Cluster-4 :  666

No. of reviews in Cluster-5 :  278

No. of reviews in Cluster-6 :  139

No. of reviews in Cluster-7 :  210
```

[5.1.4] Wordclouds of clusters obtained after applying k-means on TFIDF SET 2

In [124]:
```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
```

```
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster3))
```

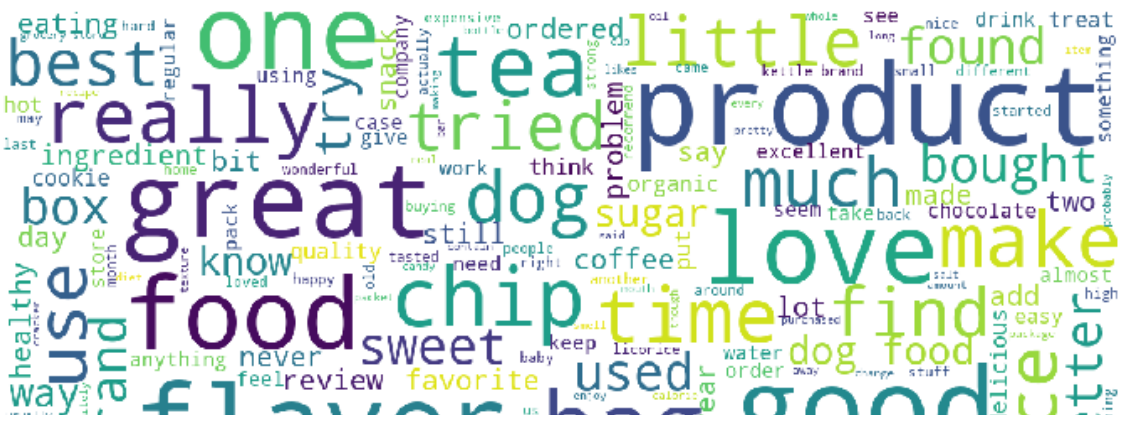
```
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 3')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster4))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 4')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster5))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 5')
plt.show()
```

```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster6))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 6')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster7))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 7')
plt.show()
```

Worldcloud for cluster 2

Worldcloud for cluster 3

Worldcloud for cluster 4



Worldcloud for cluster 5

Worldcloud for cluster 6

Worldcloud for cluster 7

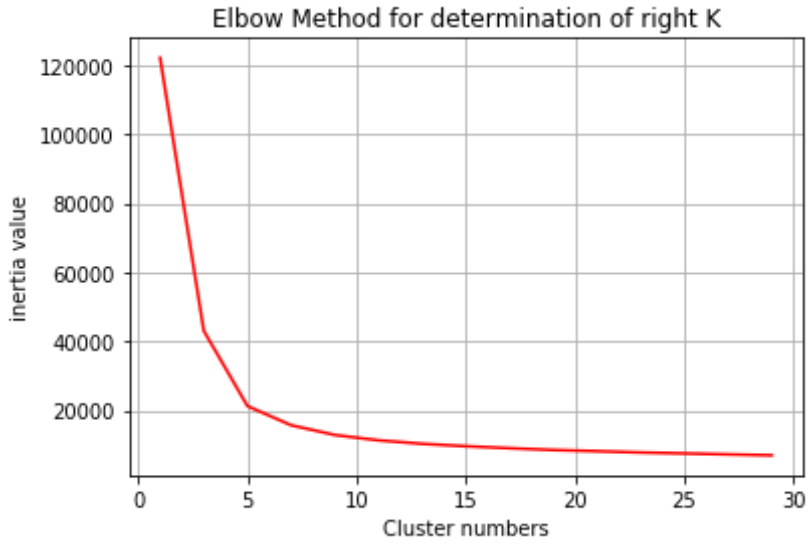
[5.1.5] Applying K-Means Clustering on AVG W2V, SET 3

In [125]:
```
# Please write all the code with proper documentation
k=list(range(1,30,2))
inertia=[]
for i in k:
    model=KMeans(n_clusters=i)
    model.fit(sent_vectors_train)
    model_result=model.inertia_
    inertia.append(model.inertia_)
    print('for the value of k=%d the value of inertia is %f'%(i,model_r
esult))
plt.plot(k,inertia,c='r')
plt.grid()
plt.xlabel('Cluster numbers')
plt.ylabel('inertia value')
plt.title('Elbow Method for determination of right K')
plt.show
```

```
for the value of k=1 the value of inertia is 122150.000000
for the value of k=3 the value of inertia is 43043.334958
for the value of k=5 the value of inertia is 21240.895715
for the value of k=7 the value of inertia is 15701.133318
for the value of k=9 the value of inertia is 12869.568734
for the value of k=11 the value of inertia is 11326.133983
for the value of k=13 the value of inertia is 10313.636257
for the value of k=15 the value of inertia is 9615.683989
for the value of k=17 the value of inertia is 9039.083915
for the value of k=19 the value of inertia is 8529.892944
for the value of k=21 the value of inertia is 8175.859212
for the value of k=23 the value of inertia is 7799.649267
for the value of k=25 the value of inertia is 7528.097810
```

```
for the value of k=27 the value of inertia is 7249.884024
for the value of k=29 the value of inertia is 6982.553685
```

Out[125]: <function matplotlib.pyplot.show(*args, **kw)>



In [126]:
```
reviews=final['Preprocessed_reviews'].values
model_new=KMeans(n_clusters=6)
model_new.fit(final_counts)
cluster1 = []
cluster2 = []
cluster3 = []
cluster4 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif model_new.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif model_new.labels_[i] == 2:
        cluster3.append(reviews[i])
    else:
        cluster4.append(reviews[i])
```

```
# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
```

```
No. of reviews in Cluster-1 :  1

No. of reviews in Cluster-2 :  12

No. of reviews in Cluster-3 :  350

No. of reviews in Cluster-4 :  2080
```

[5.1.6] Wordclouds of clusters obtained after applying k-means on AVG W2V SET 3

In [127]:
```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()
```

```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster3))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 3')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
```

```
ads
    ).generate(str(cluster4))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 4')
plt.show()
```

Worldcloud for cluster 2

Worldcloud for cluster 3

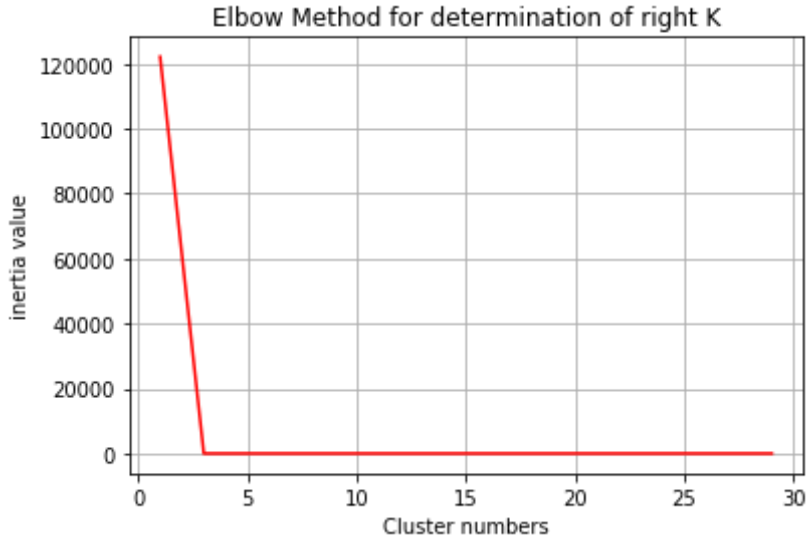
[5.1.7] Applying K-Means Clustering on TFIDF W2V, SET 4

In [92]:
```
# Please write all the code with proper documentation
k=list(range(1,30,2))
inertia=[]
for i in k:
    model=KMeans(n_clusters=i)
    model.fit(tfidf_sent_vectors_train)
    model_result=model.inertia_
    inertia.append(model.inertia_)
    print('for the value of k=%d the value of inertia is %f'%(i,model_r
esult))
plt.plot(k,inertia,c='r')
plt.grid()
plt.xlabel('Cluster numbers')
```

```
plt.ylabel('inertia value')
plt.title('Elbow Method for determination of right K')
plt.show
```

```
for the value of k=1 the value of inertia is 122150.000000
for the value of k=3 the value of inertia is 0.000000
for the value of k=5 the value of inertia is 0.000000
for the value of k=7 the value of inertia is 0.000000
for the value of k=9 the value of inertia is 0.000000
for the value of k=11 the value of inertia is 0.000000
for the value of k=13 the value of inertia is 0.000000
for the value of k=15 the value of inertia is 0.000000
for the value of k=17 the value of inertia is 0.000000
for the value of k=19 the value of inertia is 0.000000
for the value of k=21 the value of inertia is 0.000000
for the value of k=23 the value of inertia is 0.000000
for the value of k=25 the value of inertia is 0.000000
for the value of k=27 the value of inertia is 0.000000
for the value of k=29 the value of inertia is 0.000000
```

Out[92]: <function matplotlib.pyplot.show(*args, **kw)>



In [135]:
```
reviews=final['Preprocessed_reviews'].values
```

```
model_new=KMeans(n_clusters=3)
model_new.fit(tfidf_sent_vectors_train)
cluster1 = []
cluster2 = []
cluster3 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif model_new.labels_[i] == 1:
        cluster2.append(reviews[i])
    else:
        cluster3.append(reviews[i])



# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
```

```
No. of reviews in Cluster-1 :  1123

No. of reviews in Cluster-2 :  9

No. of reviews in Cluster-3 :  1311
```

[5.1.8] Wordclouds of clusters obtained after applying k-means on TFIDF W2V SET 4

In [136]:

```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
```

```
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster3))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
```

```
plt.title('Worldcloud for cluster 3')
plt.show()
```

[5.2] Agglomerative Clustering

[5.2.1] Applying Agglomerative Clustering on AVG W2V, SET 3

In [130]:
```
# Please write all the code with proper documentation
from sklearn.cluster import AgglomerativeClustering
reviews=final['Preprocessed_reviews'].values
model_new=AgglomerativeClustering(n_clusters=2)
model_new.fit(sent_vectors_train)
cluster1 = []
cluster2 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    else:
        cluster2.append(reviews[i])



# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
```

```
No. of reviews in Cluster-1 :  1512

No. of reviews in Cluster-2 :  931
```

[5.2.2] Wordclouds of clusters obtained after applying Agglomerative Clustering on AVG W2V SET 3

In [131]:
```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
```

```
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()
```

Worldcloud for cluster 1

Worldcloud for cluster 2

In [132]:
```
reviews=final['Preprocessed_reviews'].values
model_new=AgglomerativeClustering(n_clusters=5)
model_new.fit(sent_vectors_train)
cluster1 = []
cluster2 = []
cluster3=[]
cluster4=[]
cluster5=[]
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif model_new.labels_[i] == 1:
        cluster2.append(reviews[i])
    elif model_new.labels_[i] == 2:
        cluster3.append(reviews[i])
    elif model_new.labels_[i] == 3:
        cluster4.append(reviews[i])
    else:
        cluster5.append(reviews[i])



# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
print("\nNo. of reviews in Cluster-4 : ",len(cluster4))
print("\nNo. of reviews in Cluster-5 : ",len(cluster5))
```

```
No. of reviews in Cluster-1 :  1125

No. of reviews in Cluster-2 :  378

No. of reviews in Cluster-3 :  9

No. of reviews in Cluster-4 :  711
```

```
No. of reviews in Cluster-5 :  220
```

In [133]:

```
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
```

```
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster3))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 3')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster4))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 4')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster5))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')
```

```
plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 5')
plt.show()
```

Worldcloud for cluster 3

Worldcloud for cluster 4

Worldcloud for cluster 5

[5.2.3] Applying Agglomerative Clustering on TFIDF W2V, SET 4

In [137]:
```
# Please write all the code with proper documentation
reviews=final['Preprocessed_reviews'].values
model_new=AgglomerativeClustering(n_clusters=3)
model_new.fit(tfidf_sent_vectors_train)
cluster1 = []
cluster2 = []
cluster3 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    elif model_new.labels_[i] == 1:
        cluster2.append(reviews[i])
    else:
        cluster3.append(reviews[i])



# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
print("\nNo. of reviews in Cluster-3 : ",len(cluster3))
```

```
No. of reviews in Cluster-1 :  1340

No. of reviews in Cluster-2 :  9

No. of reviews in Cluster-3 :  1094
```

[5.2.4] Wordclouds of clusters obtained after applying Agglomerative Clustering on TFIDF W2V SET 4

In [138]:
```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
```

```
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster3))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 3')
plt.show()
```

Worldcloud for cluster 2

Worldcloud for cluster 3

[5.3] DBSCAN Clustering

[5.3.1] Applying DBSCAN on AVG W2V, SET 3

In [145]:
```
# Please write all the code with proper documentation
from sklearn.cluster import DBSCAN
reviews=final['Preprocessed_reviews'].values
model_new=DBSCAN(eps=0.5)
model_new.fit(sent_vectors_train)
n_clusters = len(set(model_new.labels_))
print('The number of clusters are=%d'%n_clusters)
```

```
The number of clusters are=2
```

In [146]:
```
cluster1 = []
cluster2 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    else:
        cluster2.append(reviews[i])



# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
```

```
No. of reviews in Cluster-1 :  9

No. of reviews in Cluster-2 :  2434
```

[5.3.2] Wordclouds of clusters obtained after applying DBSCAN on

AVG W2V SET 3

In [147]:
```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()
```

Worldcloud for cluster 1

Worldcloud for cluster 2

[5.3.3] Applying DBSCAN on TFIDF W2V, SET 4

In [148]:
```
# Please write all the code with proper documentation
reviews=final['Preprocessed_reviews'].values
model_new=DBSCAN(eps=0.5)
model_new.fit(tfidf_sent_vectors_train)
n_clusters = len(set(model_new.labels_))
print('The number of clusters are=%d'%n_clusters)
```

```
The number of clusters are=2
```

In [149]:
```
cluster1 = []
cluster2 = []
for i in range(model_new.labels_.shape[0]):
    if model_new.labels_[i] == 0:
        cluster1.append(reviews[i])
    else:
        cluster2.append(reviews[i])
```

```
# Number of reviews in different clusters
print("No. of reviews in Cluster-1 : ",len(cluster1))
print("\nNo. of reviews in Cluster-2 : ",len(cluster2))
```

```
No. of reviews in Cluster-1 :  2434

No. of reviews in Cluster-2 :  9
```

[5.3.4] Wordclouds of clusters obtained after applying DBSCAN on TFIDF W2V SET 4

In [150]:
```
# Please write all the code with proper documentation
wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster1))
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 1')
plt.show()

wordcloud = WordCloud(
        background_color='white',
        stopwords=stopwords,
        max_words=200,
        max_font_size=40,
        scale=3,
        random_state=1 # chosen at random by flipping a coin; it was he
ads
    ).generate(str(cluster2))
```

```
fig = plt.figure(1, figsize=(12, 12))
plt.axis('off')

plt.imshow(wordcloud)
plt.title('Worldcloud for cluster 2')
plt.show()
```

[6] Conclusions

In [152]:
```
# Please compare all your models using Prettytable library.
# You can have 3 tables, one each for kmeans, agllomerative and dbscan
from prettytable import PrettyTable
x=PrettyTable()
x.field_names=['Vectorizer','Clustering Technique','Number Of Clusters'
]
x.add_row(['BOW','K-means',6])
x.add_row(['TFIDF','K-means',7])
x.add_row(['Avg W2V','K-means',6])
x.add_row(['TFIDF weighted W2V','K-means',3])
x.add_row(['Avg W2V','Agglomerative',2])
x.add_row(['TFIDF Weighted W2V','Agglomerative',2])
x.add_row(['Avg W2V','DBSCAN',2])
```

```
x.add_row(['TFIDF Weighted W2V','DBSCAN',2])
print(x)
```

```
+--------------------+----------------------+--------------------+
|     Vectorizer     | Clustering Technique | Number Of Clusters |
+--------------------+----------------------+--------------------+
|        BOW         |       K-means        |         6          |
|       TFIDF        |       K-means        |         7          |
|      Avg W2V       |       K-means        |         6          |
| TFIDF weighted W2V |       K-means        |         3          |
|      Avg W2V       |    Agglomerative     |         2          |
| TFIDF Weighted W2V |    Agglomerative     |         2          |
|      Avg W2V       |        DBSCAN        |         2          |
| TFIDF Weighted W2V |        DBSCAN        |         2          |
+--------------------+----------------------+--------------------+
```