Data_Cleaning_AFF_Review

November 15, 2018

EDA on Amazon Fine Food Review dataset ===

1 Mount Google Drive

In [1]:
```
# Mouting Google Drive
#from google.colab import drive
#drive.mount('/content/drive')
```

2 Import Required Modules

In [2]:
```
import sqlite3
import pandas as pd
import numpy as np
import csv # for CSV file handling
#from tqdm import tqdm_notebook
from tqdm import tqdm
import re # for regular expression over sentences for pre-processing
from nltk.corpus import stopwords # for stopwords removal

import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /home/shin/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[2]: True

3 Load Data

In [3]:
```
# Using sqlite read data from the database
#con = sqlite3.connect('/content/drive/My Drive/Colab Notebooks/AFF-Review/database.sqli
#con = sqlite3.connect('./../../../../../Instructor_Notebooks/AmazonFineFoodReviews/data
con = sqlite3.connect('./../appliedaicourse/AFF-Review/database.sqlite')

# Get reviews which do not have score as 3
```

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

Out[3]:

	Id	ProductId	UserId	ProfileName	\
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	

	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	\
0	1	1	5	1303862400	
1	0	0	1	1346976000	
2	1	1	4	1219017600	
3	3	3	2	1307923200	
4	0	0	5	1350777600	

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

4 Highlevel Statistics

```
In [4]: filtered_data.describe()
```

Out[4]:

	Id	HelpfulnessNumerator	HelpfulnessDenominator	\
count	525814.000000	525814.000000	525814.000000	
mean	284599.060038	1.747293	2.209544	
std	163984.038077	7.575819	8.195329	
min	1.000000	0.000000	0.000000	
25%	142730.250000	0.000000	0.000000	
50%	284989.500000	0.000000	1.000000	
75%	426446.750000	2.000000	2.000000	
max	568454.000000	866.000000	878.000000	

	Score	Time
count	525814.000000	5.258140e+05
mean	4.279148	1.295943e+09
std	1.316725	4.828129e+07
min	1.000000	9.393408e+08
25%	4.000000	1.270598e+09
50%	5.000000	1.310861e+09
75%	5.000000	1.332634e+09
max	5.000000	1.351210e+09

4.1 Features/ Labels

```
In [5]: filtered_data.columns
```

```
Out[5]: Index(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
               'HelpfulnessDenominator', 'Score', 'Time', 'Summary', 'Text'],
              dtype='object')
```

```
In [6]: filtered_data.dtypes
```

```
Out[6]: Id                      int64
        ProductId              object
        UserId                 object
        ProfileName            object
        HelpfulnessNumerator    int64
        HelpfulnessDenominator  int64
        Score                   int64
        Time                    int64
        Summary                object
        Text                   object
        dtype: object
```

4.1.1 Observation

- Totally 10 features given
- No labels given
- From Kaggle below information I have obtained about teach feature
  - https://www.kaggle.com/snap/amazon-fine-food-reviews
- Id
  - Row Id
- ProductId
  - Unique identifier for the product
- UserId
  - Unqiue identifier for the user
- ProfileName
  - Profile name of the user
- HelpfulnessNumerator
  - Number of users who found the review helpful
- HelpfulnessDenominator
  - Number of users who indicated whether they found the review helpful
- Score

  - Rating between 1 and 5
- Time
  - Timestamp for the review
- Summary
  - Brief summary of the review
- Text
  - Text of the review

5 Data Cleaning

5.1 Analysis

5.1.1 Id

```
In [7]: u = filtered_data.Id.value_counts()
        u.unique()
```

```
Out[7]: array([1])
```

Observation

- No Id repeation

5.1.2 ProductId

```
In [8]: len(filtered_data.ProductId.unique())
```

```
Out[8]: 72005
```

Observation

- 72005 Products

5.1.3 UserId

```
In [9]: len(filtered_data.UserId.unique())
```

```
Out[9]: 243414
```

Observation

- 243414 Users

5.1.4 HelpfulnessNumerator

```
In [10]: print(filtered_data.HelpfulnessNumerator.min(),
               filtered_data.HelpfulnessNumerator.max(),
               len(filtered_data.HelpfulnessNumerator.unique()))
```

```
0 866 222
```

Observation

- value ranges from 0 to 808
- 222 unique entries

5.1.5 HelpfulnessDenominator

```
In [11]: print(filtered_data.HelpfulnessDenominator.min(),
               filtered_data.HelpfulnessDenominator.max(),
               len(filtered_data.HelpfulnessDenominator.unique()))
```

```
0 878 227
```

```
In [12]: # As per feature details, Denominator should be greater than Numerator
         # Lets check whether the data follows that description
         filtered_data[(filtered_data.HelpfulnessDenominator < filtered_data.HelpfulnessNumerato
```

Out[12]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
41159	44737	B001EQ55RW	A2V0I904FH7ABY	Ram	3	2	4	1212883200	Pure cocoa taste with crunchy almonds inside	It was almost a 'love at first bite' - the per...
59301	64422	B000MIDROQ	A161DK06JJMCYF	J. E. Stephens "Jeanne"	3	1	5	1224892800	Bought This for My Son at College	My son loves spaghetti so I didn't hesitate or...

Observation

- value ranges from 0 to 878
- 227 unique entries
- **2 invalid entries found**
  - Denominator is greater than Numerator

5.1.6 Score

```
In [13]: filtered_data.Score.unique()
```

```
Out[13]: array([5, 1, 4, 2])
```

```
In [14]: filtered_data.Score.value_counts()
```

```
Out[14]: 5    363122
         4     80655
         1     52268
         2     29769
         Name: Score, dtype: int64
```

Observation

- Scores range from 1 to 5 only
- No invalid entries found
- **No equal amount of data points for each score**
  - We have an IMBALANCED dataset

5.1.7 Time

```
In [15]: len(filtered_data.Time.unique())
```

```
Out[15]: 3157
```

```
In [16]: #filtered_data['Time'].value_counts()
```

```
In [17]: # Check whether any entry with same time for more than one product
         # which is practically not possible
         userid_group = filtered_data.groupby('UserId')
         #g = userid_group.groups
         #g.values()
```

```
In [18]: userid_group.filter(lambda x:len(x)>1).sort_values('Time')
```

```
Out[18]:             Id   ProductId          UserId             ProfileName  \
         346055  374359  B00004CI84  A344SMIA5JECGM         Vincent P. Ross
         417859  451878  B00004CXX9  A344SMIA5JECGM         Vincent P. Ross
         212472  230285  B00004RYGX  A344SMIA5JECGM         Vincent P. Ross
         346116  374422  B00004CI84  A1048CYU0OV4O8            Judy L. Eans
         417927  451949  B00004CXX9  A1048CYU0OV4O8            Judy L. Eans
         212533  230348  B00004RYGX  A1048CYU0OV4O8            Judy L. Eans
         417847  451864  B00004CXX9   A1B2IZU1JLZA6                    Wes
         212458  230269  B00004RYGX   A1B2IZU1JLZA6                    Wes
         346041  374343  B00004CI84   A1B2IZU1JLZA6                    Wes
         346141  374450  B00004CI84   ACJR7EQF9S6FP        Jeremy Robertson
         212558  230376  B00004RYGX   ACJR7EQF9S6FP        Jeremy Robertson
```

417952	451977	B00004CXX9	ACJR7EQF9S6FP	Jeremy Robertson
212511	230326	B00004RYGX	A2DEE7F9XKP3ZR	jerome
346094	374400	B00004CI84	A2DEE7F9XKP3ZR	jerome
417883	451903	B00004CXX9	A2DEE7F9XKP3ZR	jerome
138001	149770	B00004S1C5	A1KXONFPU2XQ5K	Stephanie Manley
138017	149789	B00004S1C6	A1KXONFPU2XQ5K	Stephanie Manley
212532	230347	B00004RYGX	A1FJOY14X3MUHE	Justin Howard
417926	451948	B00004CXX9	A1FJOY14X3MUHE	Justin Howard
346115	374421	B00004CI84	A1FJOY14X3MUHE	Justin Howard
346102	374408	B00004CI84	A1GB1Q193DNFGR	Bruce Lee Pullen
212519	230334	B00004RYGX	A1GB1Q193DNFGR	Bruce Lee Pullen
417913	451935	B00004CXX9	A1GB1Q193DNFGR	Bruce Lee Pullen
212495	230309	B00004RYGX	A34NBH479RB0E	"dmab6395"
346078	374383	B00004CI84	A34NBH479RB0E	"dmab6395"
417882	451902	B00004CXX9	A34NBH479RB0E	"dmab6395"
346054	374358	B00004CI84	A1HWMNSQF14MP8	will@socialaw.com
417858	451877	B00004CXX9	A1HWMNSQF14MP8	will@socialaw.com
212471	230284	B00004RYGX	A1HWMNSQF14MP8	will@socialaw.com
138018	149790	B00004S1C6	A1IU7S4HCK1XK0	Joanna Daneman
...	...	...	...	...
427278	462088	B00611F084	A6D4ND3C3BCYV	karo
218306	236653	B008YA1NWC	A2O4V3MCB7EPPU	Bellingham Bookworm
372276	402585	B000EML7DS	A2DFSA2JXQKVY3	C-Rush
280723	304160	B001AS1A4Q	A2E2F8WSUB33VE	Maria A. Alfonzo
280722	304159	B001AS1A4Q	AYTSBGA5A3UWI	Imran Ali
19181	20930	B001L1MKLY	A38XYFHXEUNUW6	bleaufire
118532	128554	B007L3NVKU	A3HM6TNYB7FNDL	C. Furman
279857	303246	B0002DGRZC	AUINI96NMGXUI	Kkrys23
279856	303245	B0002DGRZC	A3SSEJ8IEM4YGW	Seagaul
279331	302676	B000UBH9YE	A1CM50V04TUUPF	Shelly
395966	428155	B003XKF6CQ	A3IYSIAKYOMKTO	Renter
119196	129256	B004MMNNDS	A248RO4GSIWDII	Robert Kawalec
371881	402156	B0006349WQ	A21BT40VZCCYT4	Carol A. Reed
219434	237869	B003ASXKV0	AUEA2NJHMK9DF	Penny E. Cooke "PMSDEA"
219497	237940	B00018CWN4	A37264CFSSA73O	Andrea
80489	87518	B0050CPSBE	A4IL0CLL27Q33	D. Brennan
482305	521517	B002HNC8VW	A2DVFHG099GUGE	sauerkraut
393073	425059	B00317HLQA	A3AOK34N9VZ7HY	college student mom
220272	238767	B008RRJCDY	A1W6E1FN0745L7	J. Tomaszewski
50708	55049	B000IHJEDE	A2DFSA2JXQKVY3	C-Rush
350425	379063	B0000V1B3E	A3PKAVKWFFT0GC	FinGurBang
393021	424999	B0001TNCK0	A1GCFTFXELCHRP	Big Texas
366461	396260	B007FK3JS8	A11X0ENDTFGCEH	marval
183133	198643	B002AQL00G	AEWJD0G85FPSG	Cathy
277880	301125	B003Z6ZGZK	A2GW6JUVTALDPV	DL
428665	463583	B004QDA8WC	AFF6F08FRSYWG	Kentucky Woman "Emily"
317938	344192	B007S0WQXE	A2BV01F023AUW1	E. Bitterlich
509087	550476	B001SAXPE0	A32NC2UF34RJQY	D. Pagliassotti

```
184801  200465  B008O2EHNC  A11X0ENDTFGCEH                marval
491422  531341  B0002DGRSY  A3SSEJ8IEM4YGW               Seagaul

        HelpfulnessNumerator  HelpfulnessDenominator  Score        Time  \
346055                     1                       2      5   944438400
417859                     1                       2      5   944438400
212472                     1                       2      5   944438400
346116                     2                       2      5   947376000
417927                     2                       2      5   947376000
212533                     2                       2      5   947376000
417847                    19                      23      1   948240000
212458                    19                      23      1   948240000
346041                    19                      23      1   948240000
346141                     2                       3      4   951523200
212558                     2                       3      4   951523200
417952                     2                       3      4   951523200
212511                     0                       3      5   959990400
346094                     0                       3      5   959990400
417883                     0                       1      5   959990400
138001                     8                       8      5   965779200
138017                    26                      28      5   965779200
212532                     2                       2      5   966297600
417926                     2                       2      5   966297600
346115                     2                       2      5   966297600
346102                     5                       5      5   970531200
212519                     5                       5      5   970531200
417913                     5                       5      5   970531200
212495                     0                       1      5   977184000
346078                     0                       1      5   977184000
417882                     0                       1      5   977184000
346054                     1                       2      5   978134400
417858                     1                       2      5   978134400
212471                     1                       2      5   978134400
138018                    25                      27      5   982800000
...                      ...                     ...    ...         ...
427278                     0                       0      5  1351209600
218306                     0                       0      4  1351209600
372276                     0                       0      4  1351209600
280723                     0                       0      5  1351209600
280722                     0                       0      5  1351209600
19181                      0                       0      5  1351209600
118532                     0                       0      4  1351209600
279857                     0                       0      5  1351209600
279856                     0                       0      5  1351209600
279331                     0                       0      5  1351209600
395966                     0                       0      5  1351209600
119196                     0                       0      5  1351209600
371881                     0                       0      5  1351209600
```

```
219434                     0                         0      4  1351209600
219497                     0                         0      5  1351209600
80489                      0                         0      1  1351209600
482305                     0                         0      2  1351209600
393073                     0                         0      5  1351209600
220272                     0                         0      5  1351209600
50708                      0                         0      4  1351209600
350425                     0                         0      1  1351209600
393021                     0                         0      4  1351209600
366461                     0                         0      5  1351209600
183133                     0                         0      5  1351209600
277880                     0                         0      1  1351209600
428665                     0                         0      5  1351209600
317938                     0                         0      5  1351209600
509087                     0                         0      5  1351209600
184801                     0                         0      5  1351209600
491422                     0                         0      5  1351209600

                                                  Summary  \
346055                             A modern day fairy tale
417859                             A modern day fairy tale
212472                             A modern day fairy tale
346116                                               GREAT
417927                                               GREAT
212533                                               GREAT
417847    WARNING: CLAMSHELL EDITION IS EDITED TV VERSION
212458    WARNING: CLAMSHELL EDITION IS EDITED TV VERSION
346041    WARNING: CLAMSHELL EDITION IS EDITED TV VERSION
346141           Bettlejuice...Bettlejuice...BETTLEJUICE!
212558           Bettlejuice...Bettlejuice...BETTLEJUICE!
417952           Bettlejuice...Bettlejuice...BETTLEJUICE!
212511      Research - Beatlejuice video - French version
346094      Research - Beatlejuice video - French version
417883                                           Research
138001                                 Very easy to use
138017                                     A must have!
212532  A fresh, original film from master storyteller...
417926  A fresh, original film from master storyteller...
346115  A fresh, original film from master storyteller...
346102       Fabulous Comedic Fanasy Directed by a Master
212519       Fabulous Comedic Fanasy Directed by a Master
417913       Fabulous Comedic Fanasy Directed by a Master
212495                                               FUNNY
346078                                               FUNNY
417882                                               FUNNY
346054                               A Afterlife Success
417858                               A Afterlife Success
212471                               A Afterlife Success
```

```
138018  Make your own Martha Stewart style cakes and c...
...                                                   ...
427278                              Jamica Me Crazy Coffee
218306                  One of my favorite K-cups flavors
372276                                           Not bad.
280723                                           Excelent
280722                               A God Sent Remedy!!!
19181                                  Yummy & Subtle
118532         Full- bodied without a bitter after-taste
279857                                 Love this faucet
279856                                    Dogs love it.
279331                                 Love My Senseo!
395966                                           Mellow
119196                                           Love it!
371881                               Good Training Treat
219434                                    Like this tea
219497                                 Great quality!
80489                                     Buyer beware
482305                       Not a preferential hot sauce
393073       special k fruit krisps. Blueberry are great
220272                            Great Choice on Popcorn
50708                                            Not bad.
350425  Want To Pay $31.51 Lb For Loose Tea That's Med...
393021                    Still unsure about its benefits.
366461     Enjoyable, quick cups of coffee with no waste
183133  Betty Crocker Gluten Free Chocolate chip cooki...
277880                          I did not receive my order
428665      Love chai - love Keurig - love these K-cups!
317938  Exactly what you think- Olive Garden's salad d...
509087                                Great for HS lunch
184801     Enjoyable, quick cups of coffee with no waste
491422                                    Dogs love it.

                                                    Text
346055  A twist of rumplestiskin captured on film, sta...
417859  A twist of rumplestiskin captured on film, sta...
212472  A twist of rumplestiskin captured on film, sta...
346116  THIS IS ONE MOVIE THAT SHOULD BE IN YOUR MOVIE...
417927  THIS IS ONE MOVIE THAT SHOULD BE IN YOUR MOVIE...
212533  THIS IS ONE MOVIE THAT SHOULD BE IN YOUR MOVIE...
417847  I, myself always enjoyed this movie, it's very...
212458  I, myself always enjoyed this movie, it's very...
346041  I, myself always enjoyed this movie, it's very...
346141  What happens when you say his name three times...
212558  What happens when you say his name three times...
417952  What happens when you say his name three times...
212511  I'm getting crazy.I'm looking for Beatlejuice ...
346094  I'm getting crazy.I'm looking for Beatlejuice ...
```

```
417883  I'm getting crazy.<p>Is it really impossible t...
138001  This are so much easier to use than the Wilson...
138017  These are easy to use, they do not make a mess...
212532  This is such a great film, I don't even know h...
417926  This is such a great film, I don't even know h...
346115  This is such a great film, I don't even know h...
346102  Beetlejuice is an awe-inspiring wonderfully am...
212519  Beetlejuice is an awe-inspiring wonderfully am...
417913  Beetlejuice is an awe-inspiring wonderfully am...
212495  I THOUGHT THIS MOVIE WAS SO FUNNY, MICHAEL KEA...
346078  I THOUGHT THIS MOVIE WAS SO FUNNY, MICHAEL KEA...
417882  I THOUGHT THIS MOVIE WAS SO FUNNY, MICHAEL KEA...
346054  Many movies, have dealt with the figure of dea...
417858  Many movies, have dealt with the figure of dea...
212471  Many movies, have dealt with the figure of dea...
138018  I don't know why anyone would ever use those l...
...                                                   ...
427278  Wolfgang Puck's Jamaica Me Crazy is that wonde...
218306  This is one of my favorite k-cup flavors.  The...
372276  These are small and very salty. The taste is g...
280723  Good price, flavor, fast delivery And good pre...
280722  I love this stuff! It's a God sent Remedy for ...
19181   Just made my first pot of this wonderful coffe...
118532  This is my everyday coffee choice...a good all...
279857  Love this faucet.  My husband had installed th...
279856  This is the "all gone" treat after dinner.  It...
279331  I I haven't had a bad cup of coffee yet.  So f...
395966  This honey made from blueberry blossoms has a ...
119196  Heard great things about drinking this tea. I ...
371881  My dog will come in from outside when I am tra...
219434  This tea has a nice flavor although I wish it ...
219497  This product is very good and I won't change i...
80489   Nespresso makes GREAT coffee and GREAT machine...
482305  For quite some time, I have been using differe...
393073  <a href="http://www.amazon.com/gp/product/B003...
220272  This powder is unlike anything I've had with i...
50708   These are small and very salty. The taste is g...
350425  Holy cow, when I placed my order for 24 indivi...
393021  ACV is supposed to help maintain the immune sy...
366461  My mother loves this coffee and the pods fit h...
183133  The Betty Crocker Gluten Free chocolate chip c...
277880  I placed my order through Amazon and after abo...
428665  I'm addicted to these chai k-cups.  It tastes ...
317938  This salad dressing is exactly what you get wh...
509087  Great for HS lunch, kid enjoy as a snack also,...
184801  My mother loves this coffee and the pods fit h...
491422  This is the "all gone" treat after dinner.  It...
```

```
[357746 rows x 10 columns]
```

5.1.8 Invalid entries check on Summary, Text

In [19]:
```
#filtered_data[filtered_data['Summary'].str.contains('book')]
#type(filtered_data[filtered_data['Summary'].str.contains('book')].index.tolist())

#suspicious_indices = []
#
#l = filtered_data[filtered_data['Summary'].str.contains('book')].index.tolist()
#print("No. of entries having '{0}' is {1}".format('book', len(l)))
#suspicious_indices = suspicious_indices + l
#
#l = filtered_data[filtered_data['Summary'].str.contains('film')].index.tolist()
#print("No. of entries having '{0}' is {1}".format('film', len(l)))
#suspicious_indices = suspicious_indices + l
#
#l = filtered_data[filtered_data['Summary'].str.contains('Film')].index.tolist()
#print("No. of entries having '{0}' is {1}".format('Film', len(l)))
#suspicious_indices = suspicious_indices + l
#
#l = filtered_data[filtered_data['Summary'].str.contains('Book')].index.tolist()
#print("No. of entries having '{0}' is {1}".format('Book', len(l)))
#suspicious_indices = suspicious_indices + l

def getEntriesHavingTexts(df, col_to_search, text_list):
  indices = []
  counts = []
  for text in text_list:
    l = filtered_data[filtered_data[col_to_search].str.contains(text)].index.tolist()
    counts.append(len(l))
    indices = indices + l
  return indices, counts
```

In [20]:
```
text_list = ['[bB]ook']
suspicious_indices, counts = getEntriesHavingTexts(filtered_data,
                                                   'Summary',
                                                   text_list)

for i in range(len(counts)):
  print("No. of entries having '{0}' is {1}".format(text_list[i], counts[i]))

print('Total suspicious entries : ', len(suspicious_indices))
save_data = filtered_data.iloc[suspicious_indices]
save_data.to_csv('test_1.csv')
```

```
No. of entries having '[bB]ook' is 85
Total suspicious entries :  85
```

```
In [21]: text_list = ['[fF]ilm']
         suspicious_indices, counts = getEntriesHavingTexts(filtered_data,
                                                  'Summary',
                                                  text_list)

         for i in range(len(counts)):
           print("No. of entries having '{0}' is {1}".format(text_list[i], counts[i]))


         print('Total suspicious entries : ', len(suspicious_indices))
         save_data = filtered_data.iloc[suspicious_indices]
         save_data.to_csv('test_2.csv')
```

```
No. of entries having '[fF]ilm' is 24
Total suspicious entries :  24
```

```
In [22]: # Found 'Tim Burton' movies reviews in Food Reviews
         text_list = ['Tim Burton']
         suspicious_indices, counts = getEntriesHavingTexts(filtered_data,
                                                  'Summary',
                                                  text_list)

         for i in range(len(counts)):
           print("No. of entries having '{0}' is {1}".format(text_list[i], counts[i]))


         print('Total suspicious entries : ', len(suspicious_indices))
         save_data = filtered_data.iloc[suspicious_indices]
         save_data.to_csv('Tim_Burton_2.csv')
```

```
No. of entries having 'Tim Burton' is 36
Total suspicious entries :  36
```

5.1.9 Analyse for any invalid entries in review text

```
def getUniqueWords(df, col_name):
    words = set()
    #words.add(' ')
    count  = 0
    for index, row in tqdm(df.iterrows()):
        w_l = list(set(row[col_name].split()))
        words = words.union(set(w_l))
        #print(row[col_name], w_l)
        #print(list(words))
        count += 1
        #if count > 20:
        #    break
```

```
    return words

#tt = final_data[~final_data.Summary.str.isalpha()]
#print(tt.shape)
#tt.apply()

%%time
summary_words = getUniqueWords(final_data, 'Summary')

tqdm(text_words = getUniqueWords(final_data, 'Text'))

print('Total unique words in Summary: ', len(summary_words))
print('Total unique words in Review Text: ', len(text_words))

def storeSet_1(w_set, file_name):
    #csv_file = csv.writer(open(file_name), 'w')
    with open(file_name, 'w', encoding="utf-8") as csv_file:
        cw = csv.writer(csv_file)
        cw.writerow(list(w_set))

def storeSet_2(w_set, file_name):
    with open(file_name, 'w', encoding="utf-8") as csv_file:
        for w in w_set:
            csv_file.write(w)
            csv_file.write('\n')

storeSet_2(summary_words, 'summary_words.csv')
storeSet_2(text_words, 'text_words.csv')

import string

invalidChars = set(string.punctuation.replace("_", ""))

def containsAny(word, char_list):
    '''
    If any of the character in char_list found in 'word' will return True
    Otherwise returns False
    '''
    for c in char_list:
        if c in word:
            return True
    return False

def containsAll(word, char_list):
    '''
    If all of the characters in char_list found in 'word' will return True
    Otherwise returns False
    '''
    for c in char_list:
```

```
        if c not in word:
            return True
    return False

def getWordsHavingSpecialChar(df, col_name):
    words = set()
    #words.add(' ')
    count  = 0
    for index, row in df.iterrows():
        w_l = list(set(row[col_name].split()))
        w_c_l = []
        for w in w_l:
            if containsAny(w, invalidChars):
                w_c_l.append(w)
        words = words.union(set(w_c_l))
        #print(row[col_name], w_l)
        #print(list(words))
        #count += 1
        #if count > 20:
        #    break
    return words

%%time
summary_invalid_words = getWordsHavingSpecialChar(final_data, 'Summary')

%%time
text_invalid_words = getWordsHavingSpecialChar(final_data, 'Text')

print('Total unique (invalid) words in Summary: ', len(summary_invalid_words))
print('Total unique (invalid) words in Review Text: ', len(text_invalid_words))

storeSet_2(summary_invalid_words, 'summary_invalid_words.csv')
storeSet_2(text_invalid_words, 'text_invalid_words.csv')
```

Observation

- There are duplicates
  - Same user having review comments for more than one product at same timestamp which is impractical

5.2 Cleaning

5.2.1 Convert Score to Numerical Value 0/1 for negative/positive review

```
In [23]: def ScoreToReviewType(score):
           if score < 3:
             return 0
           return 1

         filtered_data.Score = filtered_data.Score.map(ScoreToReviewType)
         print(filtered_data.Score.unique())
```

```
[1 0]
```

5.2.2 Drop Duplicates

```
In [24]: # Sort the data based on ProductID in ascending order so that we can keep only one kind
         sorted_data = filtered_data.sort_values('ProductId',axis=0, ascending=True, inplace=Fal

In [25]: # keep first entry, drop remaining duplicate entries
         final_data = sorted_data.drop_duplicates(subset={'UserId','ProfileName','Time','Text'},
         print(final_data.shape)
```

```
(364173, 10)
```

5.2.3 Remove invalid Helpfull Score entries

```
In [26]: final_data = final_data[final_data.HelpfulnessNumerator <= final_data.HelpfulnessDenomi
         print(final_data.shape)
```

```
(364171, 10)
```

5.2.4 Remove Invalid Summary Entries

Remove actual film reviews

- Tim Burton (found by filtering film words and looking into data)

```
In [27]: final_data = final_data[~final_data.Summary.str.contains('Tim Burton')]
         print(final_data.shape)
```

```
(364159, 10)
```

```
In [28]: final_data = final_data[~final_data.Text.str.contains('Tim Burton')]
         print(final_data.shape)
```

```
(364106, 10)
```

5.2.5 Remove Invalid Text (Review) Entries

```
In [29]: def removeHtmlTags(sentence):
             '''
             function to remove HTML tags in the given sentence
             '''
             reg_exp = re.compile('<.*?>', )
             cleaned_text = re.sub(reg_exp, ' ', sentence)
             return cleaned_text
```

```
        def removePunctuations(sentence):
          '''
          function to remove punctuations in the given sentence
          '''
          cleaned_sentence = re.sub(r'[?|!|\'|"|#]',r'',sentence)
          cleaned_sentence = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned_sentence)
          return cleaned_sentence

        '''
        s = 'Hi I am <pr> test </pr> testing'
        removeHtmlTags(s).split()
        '''
```

```
Out[29]: "\ns = 'Hi I am <pr> test </pr> testing'\nremoveHtmlTags(s).split()\n"
```

```
In [30]: stop_words = set(stopwords.words('english')) # get stop words for English
         #print(stop)
         snow_stem = nltk.stem.SnowballStemmer('english') # get Stemmer for English
         #print(snow)
```

```
In [31]: all_positive_words = []
         all_negative_words = []
         final_review_texts = []
         df_index = 0 # for tracking the observations

         for sent in tqdm(final_data['Text'].values):
           #print('{0} ==> '.format(df_index), sent)
           sent = removeHtmlTags(sent) # remove HTML tags first
           #print('{0} ==> '.format(df_index), sent)

           filtered_words = []
           for w in sent.split():
             #print(removePunctuations(w))
             for cleaned_word in removePunctuations(w).split():
               if ((cleaned_word.isalpha()) & (len(cleaned_word) > 2)):
                 cleaned_word = cleaned_word.lower()
                 #print(cleaned_word)
                 if (cleaned_word not in stop_words):
                   s = (snow_stem.stem(cleaned_word)).encode('utf8')
                   filtered_words.append(s)
                   if (final_data['Score'].values)[df_index] == 1:
                     all_positive_words.append(s)
                   else:
                     all_negative_words.append(s)
                 else:
                   continue
               else:
```

```
            continue
        filtered_sent = b" ".join(filtered_words)
        #print(filtered_words, filtered_sent)

        final_review_texts.append(filtered_sent)

        #df_index += 1
        #if df_index > 10:
        #  break

100%|| 364106/364106 [10:04<00:00, 602.61it/s]
```

In [32]:
```
# add cleaned text as a seperate column into our final data dataframe
final_data['CleanedText'] = final_review_texts
final_data.head()
```

Out[32]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text	CleanedText
138706	150524	0006641040	ACITT7DI6IDDL	shari zychinski	0	0	1	939340800	EVERY book is educational	this witty little book makes my son laugh at l...	b'witti littl book make son laugh loud recit c...
138688	150506	0006641040	A2IW4PEEKO2ROU	Tracy	1	1	1	1194739200	Love the book, miss the hard cover version	I grew up reading these Sendak books, and watc...	b'grew read sendak book watch realli rosi movi...
138689	150507	0006641040	A1S4A3IQ2MU7V4	sally sue "sally sue"	1	1	1	1191456000	chicken soup with rice months	This is a fun way for children to learn their ...	
138690	150508	0006641040	AZGXZ2UUK6X	Catherine Hallberg "(Kate)"	1	1	1	1076025600	a good swingy rhythm for reading aloud	This is a great little book to read aloud- it ...	
138691	150509	0006641040	A3CMRKGE0P909G	Teresa	3	4	1	1018396800	A great way to learn the months	This is a book of poetry about the months of t...	

```
138689  b'fun way children learn month year learn poem...
138690  b'great littl book read nice rhythm well good ...
138691  b'book poetri month year goe month cute littl ...
```

```
In [33]: # store final data into new database
         conn = sqlite3.connect('cleaned.sqlite')
         c = conn.cursor()
         conn.text_factory = str
         final_data.to_sql('Reviews', conn, schema=None, if_exists='replace',
                           index=True, index_label=None, dtype=None)
         conn.close()
```

Colab Code

```
!ls
!pwd
!mv "/content/cleaned.sqlite" "/content/drive/My Drive/Colab Notebooks/AFF-Review/cleaned.sqlite
!ls
```

6 Observation Summary

- TO DO