In [1]:
```
import pandas as pd
import numpy as np

import re
import string
import os
os.chdir('..')
import spacy
import conda

import gensim
import gensim.corpora as corpora
from gensim.utils import simple_preprocess

import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from pprint import pprint


# libraries for visualization
import pyLDAvis
import pyLDAvis.gensim
import pickle
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\saeid\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [2]:
```
review_data=pd.read_csv("C:/Users/saeid/OneDrive/Documents/claremont/466/HW3/Reviews.csv")
print(review_data.head(2))
print(len(review_data))
print('Unique Products')
print(len(review_data.groupby('ProductId')))
print('Unique Users')
print(len(review_data.groupby('UserId')))
```

```
   Id   ProductId          UserId ProfileName  HelpfulnessNumerator  \
0   1  B001E4KFG0  A3SGXH7AUHU8GW  delmartian                     1
1   2  B00813GRG4  A1D87F6ZCVE5NK      dll pa                     0
```

```
   HelpfulnessDenominator  Score        Time                Summary  \
0                       1      5  1303862400  Good Quality Dog Food
1                       0      1  1346976000      Not as Advertised

                                                Text
0  I have bought several of the Vitality canned d...
1  Product arrived labeled as Jumbo Salted Peanut...
568454
Unique Products
74258
Unique Users
256059
```

In [3]:

```
def clean_text(text ):
    delete_dict = {sp_character: '' for sp_character in string.punctuation}
    delete_dict[' '] = ' '
    table = str.maketrans(delete_dict)
    text1 = text.translate(table)
    #print('cleaned:'+text1)
    textArr= text1.split()
    text2 = ' '.join([w for w in textArr if ( not w.isdigit() and  ( not w.isdigit() and len(w)>3))])
    return text2.lower()
```

In [4]:

```
review_data.dropna(axis = 0, how ='any',inplace=True)

review_data['Text'] = review_data['Text'].apply(clean_text)
review_data['Num_words_text'] = review_data['Text'].apply(lambda x:len(str(x).split()))

print('-------Dataset --------')
print(review_data['Score'].value_counts())
print(len(review_data))
print('-------------------------')
max_review_data_sentence_length  = review_data['Num_words_text'].max()

mask = (review_data['Num_words_text'] < 100) & (review_data['Num_words_text'] >=20)
df_short_reviews = review_data[mask]
df_sampled = df_short_reviews.groupby('Score').apply(lambda x: x.sample(n=20000)).reset_index(drop = True)

print('No of Short reviews')
print(len(df_short_reviews))
```

```
-------Dataset --------
5    363111
4     80655
```

```
1     52264
3     42638
2     29743
Name: Score, dtype: int64
568411
------------------------
No of Short reviews
373281
```

In [5]:

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
# function to remove stopwords
def remove_stopwords(text):
    textArr = text.split(' ')
    rem_text = " ".join([i for i in textArr if i not in stop_words])
    return rem_text

# remove stopwords from the text
df_sampled['Text']=df_sampled['Text'].apply(remove_stopwords)
```

In [6]:

```
nlp = spacy.load('en_core_web_md', disable=['parser', 'ner'])

def lemmatization(texts,allowed_postags=['NOUN', 'ADJ']):
       output = []
       for sent in texts:
             doc = nlp(sent)
             output.append([token.lemma_ for token in doc if token.pos_ in allowed_postags ])
       return output
```

In [7]:

```
from nltk.stem import WordNetLemmatizer
text_list=df_sampled['Text'].tolist()
print(text_list[1])
tokenized_reviews = lemmatization(text_list)
print(tokenized_reviews[1])
```

```
tamarined candy amazing doubt howeverthe quality tamarind candy sold asian store deplorable ordered boxes tamarind candy
instead received boxes tamarind candy problem unfortunately candies stale worthy condemnation date label three boxes dont
know felt like consuming candies years past best date tasting throw three boxes away hope sick eating expired tamarind ca
ndiesbr strongly recommend anyone interested buying tamarind candy retailer asian store amazon
['candy', 'amazing', 'howeverthe', 'quality', 'tamarind', 'candy', 'asian', 'store', 'deplorable', 'box', 'tamarind', 'ca
ndy', 'box', 'tamarind', 'candy', 'problem', 'stale', 'worthy', 'condemnation', 'date', 'label', 'box', 'candy', 'year',
'good', 'date', 'tasting', 'box', 'sick', 'eating', 'tamarind', 'candiesbr', 'interested', 'tamarind', 'candy', 'retaile
r', 'asian', 'store', 'amazon']
```

In [8]:
```
dictionary = corpora.Dictionary(tokenized_reviews)
doc_term_matrix = [dictionary.doc2bow(rev) for rev in tokenized_reviews]
```

In [9]:
```
# Creating the object for LDA model using gensim library
LDA = gensim.models.ldamodel.LdaModel

# Build LDA model
lda_model = LDA(corpus=doc_term_matrix, id2word=dictionary, num_topics=10, random_state=100,
                chunksize=1000, passes=50,iterations=100)
```

In [10]:
```
lda_model.print_topics()
```

Out[10]:
```
[(0,
  '0.062*"treat" + 0.050*"dog" + 0.029*"small" + 0.019*"size" + 0.016*"large" + 0.015*"tooth" + 0.014*"little" + 0.014*"p
iece" + 0.012*"pill" + 0.012*"great"'),
 (1,
  '0.047*"sugar" + 0.034*"ingredient" + 0.034*"cereal" + 0.032*"natural" + 0.023*"syrup" + 0.021*"free" + 0.020*"diet" +
0.017*"corn" + 0.017*"cake" + 0.016*"sweet"'),
 (2,
  '0.039*"flavor" + 0.028*"good" + 0.025*"taste" + 0.021*"butter" + 0.020*"sauce" + 0.018*"great" + 0.017*"peanut" + 0.01
4*"smooth" + 0.014*"product" + 0.013*"cheese"'),
 (3,
  '0.143*"coffee" + 0.027*"good" + 0.025*"flavor" + 0.020*"strong" + 0.019*"kcup" + 0.018*"bean" + 0.017*"blend" + 0.015
*"taste" + 0.013*"roast" + 0.013*"almond"'),
 (4,
  '0.039*"product" + 0.028*"amazon" + 0.025*"price" + 0.025*"good" + 0.025*"great" + 0.024*"store" + 0.020*"time" + 0.013
*"year" + 0.011*"order" + 0.011*"local"'),
 (5,
  '0.066*"water" + 0.042*"drink" + 0.027*"coconut" + 0.024*"milk" + 0.023*"protein" + 0.022*"taste" + 0.020*"energy" + 0.
017*"good" + 0.017*"juice" + 0.016*"bottle"'),
 (6,
  '0.034*"snack" + 0.034*"chip" + 0.033*"good" + 0.029*"cookie" + 0.026*"flavor" + 0.025*"great" + 0.020*"salt" + 0.015
*"taste" + 0.013*"healthy" + 0.013*"little"'),
 (7,
  '0.120*"food" + 0.021*"cat" + 0.019*"good" + 0.019*"rice" + 0.019*"chicken" + 0.015*"meal" + 0.015*"popcorn" + 0.014*"b
aby" + 0.013*"bread" + 0.012*"organic"'),
 (8,
  '0.064*"chocolate" + 0.064*"flavor" + 0.028*"sweet" + 0.026*"bar" + 0.025*"good" + 0.024*"taste" + 0.024*"candy" + 0.01
9*"vanilla" + 0.018*"milk" + 0.017*"delicious"'),
 (9,
  '0.043*"green" + 0.035*"honey" + 0.029*"tea" + 0.027*"organic" + 0.024*"ginger" + 0.020*"hair" + 0.017*"olive" + 0.016
*"brand" + 0.015*"black" + 0.012*"salmon"')]
```

In [11]:
```
import pyLDAvis
```

```
import pyLDAvis.gensim
import pyLDAvis.sklearn
pyLDAvis.enable_notebook()
```