Data Pre-processing for t-SNE on Amazon Review Dataset

December 24, 2018

1 Dataset Info and Objective for Amazon Fine Food Reviews Analysis Dataset

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

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 the Score/Rating. A rating of 4 or 5 could be cosnidered a positive review. A review of 1 or 2 could be considered negative. A review of 3 is neutral and ignored. This is an approximate and proxy way of determining the polarity (positivity/negativity) of a review.

2 Import statements

```
In [1]: %matplotlib inline
        import warnings
        warnings.filterwarnings("ignore")

        import sqlite3
        import pandas as pd
```

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

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, auc
from nltk.stem.porter import PorterStemmer

# !pip install -U gensim
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle
import re
import os
```

3 Applying pre-conditions to the dataset

Steps to access data and apply pre-conditions before any further processing. The steps are as follows:

- Use the 'database.sqlite' file and connect the db file using pandas library
- Read the db table using the connection where the 'Score' column is not equal to '3'
- Replacing the score values with 'Positive' and 'Negative' labels. Ex: Scores 1 & 2 is labbled as 'Negative' and similarly 3 & 4 are 'Positive'.

Read the comments below for precise steps.

In [ ]:
```
# Connect to the SQLite database file using SQLITE connection to read the table data.
con = sqlite3.connect('database.sqlite')



#filtering only positive and negative reviews i.e., ommitting those reviews with Score
filtered_data = pd.read_sql_query(""" SELECT * FROM Reviews WHERE Score != 3 """, con)

# Give reviews with Score>3 a positive rating and reviews with a score<3 a negative ra
```

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


#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
```

4 Printing the final shape of the dataset and first 5 values in the dataset.

```
In [2]: print(filtered_data.shape) #looking at the number of attributes and size of the data
        filtered_data.head()
```

```
(525814, 10)
```

Out[2]:

	Id	ProductId	UserId	ProfileName	HelpfulnessNumerator	HelpfulnessDenominator	Score	Time	Summary	Text
0	1	B001E4KFG0	A3SGXH7AUHU8GW	delmartian	1	1	positive	1303862400	Good Quality Dog Food	I have bought several of the Vitality canned d...
1	2	B00813GRG4	A1D87F6ZCVE5NK	dll pa	0	0	negative	1346976000	Not as Advertised	Product arrived labeled as Jumbo Salted Peanut...
2	3	B000LQOCH0	ABXLMWJIXXAIN	Natalia Corres "Natalia Corres"	1	1	positive	1219017600	"Delight" says it all	This is a confection that has been around a fe...
3	4	B000UA0QIQ	A395BORC6FGVXV	Karl	3	3	negative	1307923200	Cough Medicine	If you are looking for the secret ingredient i...
4	5	B006K2ZZ7K	A1UQRSCLF8GW1T	Michael D. Bigham "M. Wassir"	0	0	positive	1350777600	Great taffy	Great taffy at a great price. There was a wid...

5 Data Cleaning: Deduplication

```
In [3]: #Here the user "AR5J8UI46CURR" reviews are at the
        # same timestamp(caused by error or by design)
        #Eliminate the user to the avoid the inconsistent data

        display= pd.read_sql_query("""
```

```
SELECT *
FROM Reviews
WHERE Score != 3 AND UserId="AR5J8UI46CURR"
ORDER BY ProductID
""", con)
display.head()
```

Out[3]:

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

	HelpfulnessDenominator	Score	Time	Summary	Text
0	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
1	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
2	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
3	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...
4	2	5	1199577600	LOACKER QUADRATINI VANILLA WAFERS	DELICIOUS WAFERS. I FIND THAT EUROPEAN WAFERS ...

- Removing the duplicate reviews based on 'UserId','ProfileName','Time','Text' columns in the dataset and printing the final shape of the dataset

In [4]:
```
#Deduplication of entries

final=filtered_data.drop_duplicates(subset={"UserId","ProfileName","Time","Text"}, kee
final.shape
```

Out[4]: (364173, 10)

- As per the properties of 'HelpfulnessNumerator' and 'HelpfulnessDenominator' of a review 'HelpfulnessNumerator' should be less than 'HelpfulnessDenominator' and if not the case for any review it should be elminated from the final dataset

In [5]:
```
final=final[final.HelpfulnessNumerator<=final.HelpfulnessDenominator]
```

6 (a) Removing Punctuations and HTML Tags in the Review Text

- Removing HTML Tags and Punctuations in any given review text using 'cleanhtml' and 'cleanpunc' funtions; which serves no purpose when making any predictions.
- Downloading the stopwords for English language. Stopwords are words like 'and', 'do' and etc. which helps in sentence building in the language but doesn't add much value for the text processing.

```
In [6]: stop = set(stopwords.words('english')) #set of stopwords
        sno = nltk.stem.SnowballStemmer('english') #initialising the snowball stemmer

        def cleanhtml(sentence): #function to clean the word of any html-tags
            cleanr = re.compile('<.*?>')
            cleantext = re.sub(cleanr, ' ', sentence)
            return cleantext
        def cleanpunc(sentence): #function to clean the word of any punctuation or special cha
            cleaned = re.sub(r'[?|!|\'|"|#]',r'',sentence)
            cleaned = re.sub(r'[.|,|)|(|\|/]',r' ',cleaned)
            return  cleaned
```

7 (b) For loops for cleaning the Review text

- Outer 'for' loop is to iterate the over each review text in each row and remove the HTML Tags.
- Second 'for' loop is to split each sentence into list of words.
- Third 'for' loop is to remove puctuations from the split words from sentence. After removing the punctuations form the split words, check if the word is in list of stopwords, if yes remove the word.
- Later in the Third loop, stem the word to it's original form using 'PorterStemmer' method from nltk library. This gives us the group of words where HTML tags, Punctuation symbols removed. Then words are cast to lower case and encoded to 'UTF-8'.
- At the end of the first loop, join all the lists of cleaned word list into one word group 'str1' and similarly add all the 'str1's to 'final_string' list.

```
In [7]: #Code for implementing step-by-step the checks mentioned in the pre-processing phase
        # this code takes a while to run as it needs to run on 500k sentences.
        i=0
        str1=' '
        final_string=[]
        all_positive_words=[] # store words from +ve reviews here
        all_negative_words=[] # store words from -ve reviews here.
        s=''
        for sent in final['Text'].values:
            filtered_sentence=[]
            #print(sent);
            sent=cleanhtml(sent) # remove HTML tags
            for w in sent.split():
                for cleaned_words in cleanpunc(w).split():
```

```
                if((cleaned_words.isalpha()) & (len(cleaned_words)>2)):
                    if(cleaned_words.lower() not in stop):
                        s=(sno.stem(cleaned_words.lower())).encode('utf8')
                        filtered_sentence.append(s)
                        if (final['Score'].values)[i] == 'positive':
                            all_positive_words.append(s) #list of all words used to descri
                        if(final['Score'].values)[i] == 'negative':
                            all_negative_words.append(s) #list of all words used to descri
                    else:
                        continue
                else:
                    continue
            #print(filtered_sentence)
            str1 = b" ".join(filtered_sentence) #final string of cleaned words
            #print("*****************************************************************")

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

8 (c) Writing the Cleaned text to the dataset

- Store the 'final_string' list in 'CleanedText' column of the dataset and store it in the db file.
- Open the db file with pandas library and write the 'CleanedText' column to the db file

```
In [8]: final['CleanedText']=final_string #adding a column of CleanedText which displays the d
        final['CleanedText']=final['CleanedText'].str.decode("utf-8")
            # store final table into an SQlLite table for future.
        conn = sqlite3.connect('final.sqlite')
        c=conn.cursor()
        conn.text_factory = str
        final.to_sql('Reviews', conn,  schema=None, if_exists='replace', \
                     index=True, index_label=None, chunksize=None, dtype=None)
        conn.close()
```

9 Objectives of Data Pre-processing:

- Read the dataset and to identify and the issues with the data such as relation between 'HelpfulnessNumerator' and 'HelpfulnessDenominator'.
- Removing the duplicate reviews in the data to get the unique number of reviews.
- Removing the HTML tags and puctuation symbols and the stopwords from the review text.
- Stemming the words in the 'CleanedText' column to the get the original form of the words in the column.
- Finally adding the 'CleanedText' column to the original db file for further processing.

```
In [ ]:
```