Conversion of AFR data into IMDB format

1. Objective:

```
                    To convert the Amazon fine food dataset into IMDB datset fo
   rmat.
```

In [1]:

```
import os
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import warnings
import sqlite3
warnings.filterwarnings("ignore")
```

2. Data Cleaning

In [2]:

```
#connecting database

con=sqlite3.connect("database.sqlite")

# Read data from database

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

# Removal of Duplicates

pre_data=raw_data.drop_duplicates(['UserId','ProfileName','Time','Text'],keep="firs

# Removal of Unconditioning data (denominator>numerator)

pre_data=pre_data[pre_data.HelpfulnessNumerator<=pre_data.HelpfulnessDenominator]


# Finding NaN values in dataframe

# Reference
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.isnull.html

# Findind NaN values

if pre_data.isnull().values.any() == False:
    print("There is No NaN values in the DataFrame")
else:
    print(" There is NaN values present in the DataFrame")
```

```
There is No NaN values in the DataFrame
```

In [3]:

```
# sort data based on Time

filter_data=pre_data.sort_values(by=["Time"],axis=0)

# Class Label changing
# positive class label = 1
# negative class label = 0
a=[]
for i in filter_data["Score"]:
    if i > 3:
        a.append(1)
    else:
        a.append(0)
filter_data["Score"]=a
```

In [4]:

```
filter_data.shape
```

Out[4]:

```
(364171, 10)
```

In [5]:

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

Out[5]:

```
1    307061
0     57110
Name: Score, dtype: int64
```

3. Text Preprocessing

- We took the Text column for the further review idendification task, because text is the most important feature compared to other features.

In [6]:

```
# References
# https://medium.com/@jorlugaqui/how-to-strip-html-tags-from-a-string-in-python-7cb
# https://stackoverflow.com/a/40823105/4084039
# https://stackoverflow.com/questions/19790188/expanding-english-language-contracti
# https://stackoverflow.com/questions/18082130/python-regex-to-remove-all-words-whi
# https://stackoverflow.com/questions/5843518/remove-all-special-characters-punctua
# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://gist.github.com/sebleier/554280
# stemming tutorial: https://www.geeksforgeeks.org/python-stemming-words-with-nltk/
# Lemmatisation tutorial: https://www.geeksforgeeks.org/python-lemmatization-with-n
# NLTK Stemming package list: https://www.nltk.org/api/nltk.stem.html

from nltk.stem.snowball import EnglishStemmer
import re
from tqdm import tqdm
stemmer=EnglishStemmer()
```

In [7]:

```
raw_text_data=filter_data["Text"].values
```

In [8]:

```
# Stopwords

stopwords= set(['since','br', 'the', 'i', 'me', 'my', 'myself', 'we', 'our', 'ours'
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'hi
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that',
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because',
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'thro
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all',
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', '
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've",
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', '
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't"
            'won', "won't", 'wouldn', "wouldn't"])

# expanding contractions

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 [9]:

```
preprocessed_text_data=[]
for i in tqdm(raw_text_data):
# removing of HTML tags
    a=re.sub("<.*?>"," ",i)
# removing url
    b=re.sub(r"http\S+"," ",a)
# expanding contractions
    c=decontracted(b)
# removing alpha_numeric
    d=re.sub("\S*\d\S*", " ",c)
# removing Special characters
    e=re.sub('[^A-Za-z0-9]+', ' ',d)
# removing stopwords
    k=[]
    for w in e.split():
        if w.lower() not in stopwords:
            s=(stemmer.stem(w.lower())).encode('utf8')
            k.append(s)
    preprocessed_text_data.append(b' '.join(k).decode())
```

```
100%|██████████| 364171/364171 [07:12<00:00, 841.52it/s]
```

In [10]:

```
filter_data["Text"]=preprocessed_text_data
```

In [11]:

```
filter_data.shape
```

Out[11]:

```
(364171, 10)
```

4. Conversion of Data into IMDB format

4.1 Data splitting

In [12]:

```
# Reference
# https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text
# https://github.com/PushpendraSinghChauhan/Amazon-Fine-Food-Reviews/blob/master

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
```

In [13]:

```
x=filter_data.Text
y=filter_data.Score
```

In [14]:

```
# Data Splitting

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

In [15]:

```
print(x_train.shape)
print(x_test.shape)
```

```
(291336,)
(72835,)
```

4.2 Getting vocabulary

In [16]:

```
bow=CountVectorizer()
bow.fit(x_train)
```

Out[16]:

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='conten
t',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

In [17]:

```
# Getting vocabulary

vocabulary=bow.get_feature_names()
```

In [18]:

```
len(vocabulary)
```

Out[18]:

```
64868
```

4.3. Frequency of each word

In [19]:

```
# getting word frequency and indexing

word_list=dict()
index=0
for i in tqdm(x_train.values):
    for j in i.split():
        word_list.setdefault(j,[])
        word_list[j].append(index)
        index+=1
```

```
100%|██████████| 291336/291336 [00:11<00:00, 26316.91it/s]
```

In [20]:

```
len(word_list)
```

Out[20]:

```
64887
```

In [21]:

```
# getting word frequency length

word_freq=[]
for i in tqdm(vocabulary):
    word_freq.append(len(word_list[i]))
```

```
100%|██████████| 64868/64868 [00:00<00:00, 469389.71it/s]
```

In [22]:

```
word_freq=np.asarray(word_freq)
```

In [23]:

```
# Sort by Frequency

freq_index=np.argsort(word_freq)[::-1]
```

In [24]:

```
freq_index
```

Out[24]:

```
array([38986, 32656, 56247, ..., 35021, 35017, 32433])
```

In [25]:

```
# Rank as per the frequency

freq_rank=dict()
rank=1
for i in freq_index:
    freq_rank[vocabulary[i]] = rank
    rank +=1
```

In [26]:

```
# each word into rank conversion (train_data)

X_train=[]
for i in tqdm(x_train.values):
    row_data=[]
    for j in i.split():
        if (len(j)>1):
            row_data.append(freq_rank[j])
    X_train.append(row_data)
```

```
100%|██████████| 291336/291336 [00:09<00:00, 29802.35it/s]
```

In [27]:

```
# test data

X_test=[]
for i in tqdm(x_test.values):
    row_data=[]
    for j in i.split():
        try:
            if (len(j)>1):
                row_data.append(freq_rank[j])
        except KeyError:
            row_data.append(0)
    X_test.append(row_data)
```

```
100%|██████████| 72835/72835 [00:02<00:00, 35703.65it/s]
```

In [30]:

```
print(len(X_train))
print(len(X_test))
```

```
291336
72835
```

In [34]:

```
# length of the document 1

print(len(X_train[0]))
print(len(X_test[0]))
```

```
69
31
```

In [35]:

```
# Train document values
print("word review")
print("="*100)
print(x_train.values[0])

print("Corresponding Rank of review")
print("="*100)
print(X_train[0])

# Test

print("word review")
print("="*100)
print(x_test.values[0])

print("Corresponding Rank of review")
print("="*100)
print(X_test[0])
```

```
word review
====================================================================================================
tri steer famili organ side food chain resist complain often not give
often mac n chees anni sat shelf ignor donat number version took chan
c order case price good amazon case got eaten month fall back dinner
hubbi make kid not around yesterday husband pull left kraft box made
regret told not hold candl stuff go figur think speechless minut kraf
t torch noth els compar flavor came back buy sold
Corresponding Rank of review
====================================================================================================
[11, 3643, 187, 102, 341, 17, 1762, 1903, 1081, 419, 1, 55, 419, 122
2, 225, 1770, 2082, 792, 2096, 2447, 746, 369, 352, 858, 23, 209, 26,
5, 24, 209, 90, 580, 127, 849, 114, 557, 1978, 16, 186, 1, 196, 1646,
259, 975, 452, 1367, 41, 59, 1460, 638, 1, 603, 4355, 96, 38, 624, 6
0, 11423, 237, 1367, 11973, 265, 345, 273, 4, 207, 114, 19, 562]
word review
====================================================================================================
littl sweet side not tast like strong vodka usual enjoy dessert drink
still good strong vodka sweet flavor kill alcohol tast result smooth
drink usual mix littl bit ice water perfect
Corresponding Rank of review
====================================================================================================
[29, 46, 341, 1, 3, 2, 151, 2216, 178, 66, 833, 33, 84, 5, 151, 2216,
46, 4, 1328, 1263, 3, 411, 284, 33, 178, 42, 29, 78, 204, 45, 86]
```

In [36]:

```
# Type of the file

type(X_train[0])
```

Out[36]:

```
list
```

In [37]:

```
len(y_train)
```

Out[37]:

```
291336
```

In [38]:

```
len(y_test)
```

Out[38]:

```
72835
```

In [39]:

```
y_train.value_counts()
```

Out[39]:

```
1    245654
0     45682
Name: Score, dtype: int64
```

In [40]:

```
y_test.value_counts()
```

Out[40]:

```
1    61407
0    11428
Name: Score, dtype: int64
```

5. Data export

5.1 Exporting of data in pickle file

In [42]:

```
# https://www.programiz.com/python-programming/working-csv-files
# https://www.jessicayung.com/how-to-use-pickle-to-save-and-load-variables-in-pytho
# https://www.programcreek.com/python/example/99451/sklearn.externals.joblib.dump

import pickle
from sklearn.externals import joblib
```

In [43]:

```
# Train data

joblib.dump(X_train,"x_train.pkl")
```

Out[43]:

```
['x_train.pkl']
```

In [44]:

```
# Test data

joblib.dump(X_test,"x_test.pkl")
```

Out[44]:

```
['x_test.pkl']
```

In [45]:

```
# Train label

joblib.dump(y_train,"y_train.pkl")
```

Out[45]:

```
['y_train.pkl']
```

In [46]:

```
# Test label

joblib.dump(y_test,"y_test.pkl")
```

Out[46]:

```
['y_test.pkl']
```

6. Conclusion:

Step 1: Text preprocessing of Amazon fine food review(AFR) dataset was completed by using typical methods like stemming, stop words,tag removal using regular expression and many other.

Step 2: The AFR dataset was converted into IMDB dataset format by using frequency of words.

Step 3: After the conversion of imdb dataset format,the data's stored in the format of pickle for the further classification process using LSTM at google colab.