HW1-CSCI544

September 9, 2021

[1]:
```
import pandas as pd
import numpy as np
import nltk
nltk.download('wordnet')
#nltk.download('punkt')
import re
from bs4 import BeautifulSoup
import contractions
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/yingpeng/nltk_data…
[nltk_data]   Package wordnet is already up-to-date!
```

[2]:
```
! pip install bs4 # in case you don't have it installed

# Dataset: https://s3.amazonaws.com/amazon-reviews-pds/tsv/
 ↪amazon_reviews_us_Kitchen_v1_00.tsv.gz

!pip install contractions
```

```
Requirement already satisfied: bs4 in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (0.0.1)
Requirement already satisfied: beautifulsoup4 in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (from bs4) (4.9.3)
Requirement already satisfied: soupsieve>1.2 in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (from
beautifulsoup4->bs4) (2.2.1)
Requirement already satisfied: contractions in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (0.0.52)
Requirement already satisfied: textsearch>=0.0.21 in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (from contractions)
(0.0.21)
Requirement already satisfied: anyascii in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (from
textsearch>=0.0.21->contractions) (0.3.0)
Requirement already satisfied: pyahocorasick in
/Users/yingpeng/opt/anaconda3/lib/python3.8/site-packages (from
```

```
textsearch>=0.0.21->contractions) (1.4.2)
```

0.1 Read Data

[3]:
```
data = pd.read_csv("/Users/yingpeng/Downloads/HW1(1)/
↪amazon_reviews_us_Kitchen_v1_00.tsv", sep='\t',error_bad_lines = False)
```

```
b'Skipping line 16148: expected 15 fields, saw 22\nSkipping line 20100: expected
15 fields, saw 22\nSkipping line 45178: expected 15 fields, saw 22\nSkipping
line 48700: expected 15 fields, saw 22\nSkipping line 63331: expected 15 fields,
saw 22\n'
b'Skipping line 86053: expected 15 fields, saw 22\nSkipping line 88858: expected
15 fields, saw 22\nSkipping line 115017: expected 15 fields, saw 22\n'
b'Skipping line 137366: expected 15 fields, saw 22\nSkipping line 139110:
expected 15 fields, saw 22\nSkipping line 165540: expected 15 fields, saw
22\nSkipping line 171813: expected 15 fields, saw 22\n'
b'Skipping line 203723: expected 15 fields, saw 22\nSkipping line 209366:
expected 15 fields, saw 22\nSkipping line 211310: expected 15 fields, saw
22\nSkipping line 246351: expected 15 fields, saw 22\nSkipping line 252364:
expected 15 fields, saw 22\n'
b'Skipping line 267003: expected 15 fields, saw 22\nSkipping line 268957:
expected 15 fields, saw 22\nSkipping line 303336: expected 15 fields, saw
22\nSkipping line 306021: expected 15 fields, saw 22\nSkipping line 311569:
expected 15 fields, saw 22\nSkipping line 316767: expected 15 fields, saw
22\nSkipping line 324009: expected 15 fields, saw 22\n'
b'Skipping line 359107: expected 15 fields, saw 22\nSkipping line 368367:
expected 15 fields, saw 22\nSkipping line 381180: expected 15 fields, saw
22\nSkipping line 390453: expected 15 fields, saw 22\n'
b'Skipping line 412243: expected 15 fields, saw 22\nSkipping line 419342:
expected 15 fields, saw 22\nSkipping line 457388: expected 15 fields, saw 22\n'
b'Skipping line 459935: expected 15 fields, saw 22\nSkipping line 460167:
expected 15 fields, saw 22\nSkipping line 466460: expected 15 fields, saw
22\nSkipping line 500314: expected 15 fields, saw 22\nSkipping line 500339:
expected 15 fields, saw 22\nSkipping line 505396: expected 15 fields, saw
22\nSkipping line 507760: expected 15 fields, saw 22\nSkipping line 513626:
expected 15 fields, saw 22\n'
b'Skipping line 527638: expected 15 fields, saw 22\nSkipping line 534209:
expected 15 fields, saw 22\nSkipping line 535687: expected 15 fields, saw
22\nSkipping line 547671: expected 15 fields, saw 22\nSkipping line 549054:
expected 15 fields, saw 22\n'
b'Skipping line 599929: expected 15 fields, saw 22\nSkipping line 604776:
expected 15 fields, saw 22\nSkipping line 609937: expected 15 fields, saw
22\nSkipping line 632059: expected 15 fields, saw 22\nSkipping line 638546:
expected 15 fields, saw 22\n'
b'Skipping line 665017: expected 15 fields, saw 22\nSkipping line 677680:
expected 15 fields, saw 22\nSkipping line 684370: expected 15 fields, saw
22\nSkipping line 720217: expected 15 fields, saw 29\n'
b'Skipping line 723240: expected 15 fields, saw 22\nSkipping line 723433:
```

expected 15 fields, saw 22\nSkipping line 763891: expected 15 fields, saw 22\n'
b'Skipping line 800288: expected 15 fields, saw 22\nSkipping line 802942:
expected 15 fields, saw 22\nSkipping line 803379: expected 15 fields, saw
22\nSkipping line 805122: expected 15 fields, saw 22\nSkipping line 821899:
expected 15 fields, saw 22\nSkipping line 831707: expected 15 fields, saw
22\nSkipping line 842829: expected 15 fields, saw 22\nSkipping line 843604:
expected 15 fields, saw 22\n'
b'Skipping line 863904: expected 15 fields, saw 22\nSkipping line 875655:
expected 15 fields, saw 22\nSkipping line 886796: expected 15 fields, saw
22\nSkipping line 892299: expected 15 fields, saw 22\nSkipping line 902518:
expected 15 fields, saw 22\nSkipping line 903079: expected 15 fields, saw
22\nSkipping line 912678: expected 15 fields, saw 22\n'
b'Skipping line 932953: expected 15 fields, saw 22\nSkipping line 936838:
expected 15 fields, saw 22\nSkipping line 937177: expected 15 fields, saw
22\nSkipping line 947695: expected 15 fields, saw 22\nSkipping line 960713:
expected 15 fields, saw 22\nSkipping line 965225: expected 15 fields, saw
22\nSkipping line 980776: expected 15 fields, saw 22\n'
b'Skipping line 999318: expected 15 fields, saw 22\nSkipping line 1007247:
expected 15 fields, saw 22\nSkipping line 1015987: expected 15 fields, saw
22\nSkipping line 1018984: expected 15 fields, saw 22\nSkipping line 1028671:
expected 15 fields, saw 22\n'
b'Skipping line 1063360: expected 15 fields, saw 22\nSkipping line 1066195:
expected 15 fields, saw 22\nSkipping line 1066578: expected 15 fields, saw
22\nSkipping line 1066869: expected 15 fields, saw 22\nSkipping line 1068809:
expected 15 fields, saw 22\nSkipping line 1069505: expected 15 fields, saw
22\nSkipping line 1087983: expected 15 fields, saw 22\nSkipping line 1108184:
expected 15 fields, saw 22\n'
b'Skipping line 1118137: expected 15 fields, saw 22\nSkipping line 1142723:
expected 15 fields, saw 22\nSkipping line 1152492: expected 15 fields, saw
22\nSkipping line 1156947: expected 15 fields, saw 22\nSkipping line 1172563:
expected 15 fields, saw 22\n'
b'Skipping line 1209254: expected 15 fields, saw 22\nSkipping line 1212966:
expected 15 fields, saw 22\nSkipping line 1236533: expected 15 fields, saw
22\nSkipping line 1237598: expected 15 fields, saw 22\n'
b'Skipping line 1273825: expected 15 fields, saw 22\nSkipping line 1277898:
expected 15 fields, saw 22\nSkipping line 1283654: expected 15 fields, saw
22\nSkipping line 1286023: expected 15 fields, saw 22\nSkipping line 1302038:
expected 15 fields, saw 22\nSkipping line 1305179: expected 15 fields, saw 22\n'
b'Skipping line 1326022: expected 15 fields, saw 22\nSkipping line 1338120:
expected 15 fields, saw 22\nSkipping line 1338503: expected 15 fields, saw
22\nSkipping line 1338849: expected 15 fields, saw 22\nSkipping line 1341513:
expected 15 fields, saw 22\nSkipping line 1346493: expected 15 fields, saw
22\nSkipping line 1373127: expected 15 fields, saw 22\n'
b'Skipping line 1389508: expected 15 fields, saw 22\nSkipping line 1413951:
expected 15 fields, saw 22\nSkipping line 1433626: expected 15 fields, saw 22\n'
b'Skipping line 1442698: expected 15 fields, saw 22\nSkipping line 1472982:
expected 15 fields, saw 22\nSkipping line 1482282: expected 15 fields, saw
22\nSkipping line 1487808: expected 15 fields, saw 22\nSkipping line 1500636:

expected 15 fields, saw 22\n'
b'Skipping line 1511479: expected 15 fields, saw 22\nSkipping line 1532302:
expected 15 fields, saw 22\nSkipping line 1537952: expected 15 fields, saw
22\nSkipping line 1539951: expected 15 fields, saw 22\nSkipping line 1541020:
expected 15 fields, saw 22\n'
b'Skipping line 1594217: expected 15 fields, saw 22\nSkipping line 1612264:
expected 15 fields, saw 22\nSkipping line 1615907: expected 15 fields, saw
22\nSkipping line 1621859: expected 15 fields, saw 22\n'
b'Skipping line 1653542: expected 15 fields, saw 22\nSkipping line 1671537:
expected 15 fields, saw 22\nSkipping line 1672879: expected 15 fields, saw
22\nSkipping line 1674523: expected 15 fields, saw 22\nSkipping line 1677355:
expected 15 fields, saw 22\nSkipping line 1703907: expected 15 fields, saw 22\n'
b'Skipping line 1713046: expected 15 fields, saw 22\nSkipping line 1722982:
expected 15 fields, saw 22\nSkipping line 1727290: expected 15 fields, saw
22\nSkipping line 1744482: expected 15 fields, saw 22\n'
b'Skipping line 1803858: expected 15 fields, saw 22\nSkipping line 1810069:
expected 15 fields, saw 22\nSkipping line 1829751: expected 15 fields, saw
22\nSkipping line 1831699: expected 15 fields, saw 22\n'
b'Skipping line 1863131: expected 15 fields, saw 22\nSkipping line 1867917:
expected 15 fields, saw 22\nSkipping line 1874790: expected 15 fields, saw
22\nSkipping line 1879952: expected 15 fields, saw 22\nSkipping line 1880501:
expected 15 fields, saw 22\nSkipping line 1886655: expected 15 fields, saw
22\nSkipping line 1887888: expected 15 fields, saw 22\nSkipping line 1894286:
expected 15 fields, saw 22\nSkipping line 1895400: expected 15 fields, saw 22\n'
b'Skipping line 1904040: expected 15 fields, saw 22\nSkipping line 1907604:
expected 15 fields, saw 22\nSkipping line 1915739: expected 15 fields, saw
22\nSkipping line 1921514: expected 15 fields, saw 22\nSkipping line 1939428:
expected 15 fields, saw 22\nSkipping line 1944342: expected 15 fields, saw
22\nSkipping line 1949699: expected 15 fields, saw 22\nSkipping line 1961872:
expected 15 fields, saw 22\n'
b'Skipping line 1968846: expected 15 fields, saw 22\nSkipping line 1999941:
expected 15 fields, saw 22\nSkipping line 2001492: expected 15 fields, saw
22\nSkipping line 2011204: expected 15 fields, saw 22\nSkipping line 2025295:
expected 15 fields, saw 22\n'
b'Skipping line 2041266: expected 15 fields, saw 22\nSkipping line 2073314:
expected 15 fields, saw 22\nSkipping line 2080133: expected 15 fields, saw
22\nSkipping line 2088521: expected 15 fields, saw 22\n'
b'Skipping line 2103490: expected 15 fields, saw 22\nSkipping line 2115278:
expected 15 fields, saw 22\nSkipping line 2153174: expected 15 fields, saw
22\nSkipping line 2161731: expected 15 fields, saw 22\n'
b'Skipping line 2165250: expected 15 fields, saw 22\nSkipping line 2175132:
expected 15 fields, saw 22\nSkipping line 2206817: expected 15 fields, saw
22\nSkipping line 2215848: expected 15 fields, saw 22\nSkipping line 2223811:
expected 15 fields, saw 22\n'
b'Skipping line 2257265: expected 15 fields, saw 22\nSkipping line 2259163:
expected 15 fields, saw 22\nSkipping line 2263291: expected 15 fields, saw 22\n'
b'Skipping line 2301943: expected 15 fields, saw 22\nSkipping line 2304371:
expected 15 fields, saw 22\nSkipping line 2306015: expected 15 fields, saw

```
22\nSkipping line 2312186: expected 15 fields, saw 22\nSkipping line 2314740:
expected 15 fields, saw 22\nSkipping line 2317754: expected 15 fields, saw 22\n'
b'Skipping line 2383514: expected 15 fields, saw 22\n'
b'Skipping line 2449763: expected 15 fields, saw 22\n'
b'Skipping line 2589323: expected 15 fields, saw 22\n'
b'Skipping line 2775036: expected 15 fields, saw 22\n'
b'Skipping line 2935174: expected 15 fields, saw 22\n'
b'Skipping line 3078830: expected 15 fields, saw 22\n'
b'Skipping line 3123091: expected 15 fields, saw 22\n'
b'Skipping line 3185533: expected 15 fields, saw 22\n'
b'Skipping line 4150395: expected 15 fields, saw 22\n'
b'Skipping line 4748401: expected 15 fields, saw 22\n'
```

0.2 Keep Reviews and Ratings

[4]:
```
data_pp1 = data[['star_rating','review_body'] ]

#print(data_pp1.shape)
#Now I need to remove those rows without review

data_pp1_s1 = data_pp1.copy()
nan_value = float('NaN')

data_pp1_s1.replace("", nan_value, inplace = True)

data_pp1_s2 = data_pp1_s1.copy()
data_pp1_s2.dropna(subset = ['review_body'], inplace = True)

data_p1 = data_pp1_s2.copy()
#print(data_p1.shape)
```

1 Three samples reviews

[5]:
```
three_sample_review = data_p1.sample(n = 3, replace = False, random_state = 1)

pd.options.display.max_seq_items = 2000

print("First Sample")
print("star_rating: ")
print(three_sample_review.iloc[0,0])
print("review_body: ")
print(three_sample_review.iloc[0,1])


print("Second Sample")
print("star_rating: ")
```

```
print(three_sample_review.iloc[1,0])
print("review_body: ")
print(three_sample_review.iloc[1,1])


print("Third Sample")
print("star_rating: ")
print(three_sample_review.iloc[2,0])
print("review_body: ")
print(three_sample_review.iloc[2,1])
```

```
First Sample
star_rating:
4.0
review_body:
We have only used the mixer once since the Holidays but it performed quite well
and I was very happy with the gift.
Second Sample
star_rating:
5.0
review_body:
I bought one of the Blue and one of the Green bottles, since I already have the
orange one from my blender and don't care for pink. Both fit on the blade cap
with no problems.
Third Sample
star_rating:
5.0
review_body:
This is such an excellent piece of equipment. Bought initially to bake sourdough
bread, I have found so many other uses for it, too many to list. With all due
respect to Amazon and their wonderful service, I cannot understand why there
isn't a single retail business in Australia that stocks them.
```

2 Statistics of ratings before relabel

[6]:
```
data_rating1 =  data_p1[data_p1['star_rating'] == 1]
data_rating2 =  data_p1[data_p1['star_rating'] == 2]
data_rating3 =  data_p1[data_p1['star_rating'] == 3]
data_rating4 =  data_p1[data_p1['star_rating'] == 4]
data_rating5 =  data_p1[data_p1['star_rating'] == 5]

rating1_size = data_rating1.shape[0]
rating2_size = data_rating2.shape[0]
rating3_size = data_rating3.shape[0]
rating4_size = data_rating4.shape[0]
rating5_size = data_rating5.shape[0]
```

```
total_size = data_p1.shape[0]


print("Before we relabe, we have 5 ratings and their size is as below")
print("# of rating 1, # of rating 2, # of rating 3, # of rating 4, # of rating␣
 ↪5")
print(rating1_size, rating2_size, rating3_size, rating4_size, rating5_size,␣
 ↪sep=", ")

print(" percentage of rating 1, percentage of rating 2, percentage of rating 3,␣
 ↪percentage of rating 4, percentage of rating 5")
print(rating1_size/total_size, rating2_size/total_size, rating3_size/
 ↪total_size, rating4_size/total_size, rating5_size/total_size, sep=", ")
```

```
Before we relabe, we have 5 ratings and their size is as below
# of rating 1, # of rating 2, # of rating 3, # of rating 4, # of rating 5
426870, 241939, 349539, 731701, 3124595
 percentage of rating 1, percentage of rating 2, percentage of rating 3,
percentage of rating 4, percentage of rating 5
0.08756947173988501, 0.04963213723915018, 0.07170554403562598,
0.15010347422293813, 0.6409893727624008
```

3 Labelling Reviews:

3.1 The reviews with rating 4,5 are labelled to be 1 and 1,2 are labelled as 0. Discard the reviews with rating 3'

[7]:
```
#Discard the reviw with rating 3
data_p2 = data_p1[data_p1['star_rating'] != 3]
data_p3 = data_p2.copy()

#chagne label to class 1 and 0
data_p3.loc[data_p3['star_rating'] < 3, 'star_rating' ] = 0
data_p3.loc[data_p3['star_rating'] > 3 , 'star_rating'] = 1

#The neural review
data_neural = data_p1[data_p1['star_rating'] == 3]


#Positve Review
rawdata_positive = data_p3[data_p3['star_rating'] == 1]

#Negative Review
rawdata_negative = data_p3[data_p3['star_rating'] == 0]
```

4 # Statistics of ratings after relabel

[8]:
```
class1_size = rawdata_positive.shape[0]
class0_size = rawdata_negative.shape[0]
netural_size = data_neural.shape[0]

print("After we relabe, we have 3 labels")
print("# of Class 1, # of Class 0, # of neutral")
print(class1_size, class0_size, netural_size, sep=", ")

print(" percentage of Class 1, percentage of Class 0, percentage of neutral")
print(class1_size/total_size, class0_size/total_size, netural_size/total_size,␣
 ↪sep=", ")
```

```
After we relabe, we have 3 labels
# of Class 1, # of Class 0, # of neutral
3856296, 668809, 349539
 percentage of Class 1, percentage of Class 0, percentage of neutral
0.7910928469853389, 0.1372016089790352, 0.07170554403562598
```

We select 200000 reviews randomly with 100,000 positive and 100,000 negative reviews.

[9]:
```
#Select 100,000 positive and negative
data_positive = rawdata_positive.sample(n = 100000, replace = False,␣
 ↪random_state = 1)
data_negative = rawdata_negative.sample(n = 100000, replace = False,␣
 ↪random_state = 1)

#Merge them into one dataframe
data_o1 = [data_positive, data_negative]
data_overall = pd.concat(data_o1)
```

5 Before Data Cleaning, Information needed to be printed

[10]:
```
avg_reviewlenbefore = data_overall['review_body'].apply(len).mean()

print("Average length of reviews before data cleaning is "+␣
 ↪str(avg_reviewlenbefore))
```

```
Average length of reviews before data cleaning is 323.274835
```

[11]:
```
print("Three sample reviews before data cleaning")

sample3_review_bp = data_overall.sample(n = 3, replace = False, random_state =␣
 ↪3)

print("First Sample")
```

```
print(sample3_review_bp.iloc[0,1])

print("Second Sample")
print(sample3_review_bp.iloc[1,1])

print("Third Sample")
print(sample3_review_bp.iloc[2,1])
```

```
Three sample reviews before data cleaning
First Sample
These gaskets are way too thin for the magic bullet. The liquids splashes out of
the cup. I tried placing two gaskets to see if it would seal but it doesn't
work. Do not waste your money on these gasket, they do not fit the magic bullet.
Second Sample
This toaster oven worked well while it lasted, but it recently just suddenly
stopped working while it was in use and will not turn on again.  I cannot
recommend a toaster oven that only lasts for 6 months of light use in a single-
person household!
Third Sample
I barely Leave feedback on items i buy. Not because i dont want to but because i
cant on the Mobile app. But this item pissed me off when i received it, i turned
on the laptop just to give my irritated 2 cents.I received the mug in and i was
irritated and disappointed on the condition the mug was in. It appeared to have
been broken or badly chipped and they glued the piece back on and repainted it.
you can clearly see the section that has a different shade of paint.There is
glue on the inside of the mug i cannot get off.  And i thought… what company
or person would honestly sell items like this in this condition.i Guess he is
going to have to use it to hold pens at his office. Worthless.
```

6 Data Cleaning

6.1 Convert the all reviews into the lower case.

[12]:
```
data_overall['review_body'] = data_overall['review_body'].str.lower()


#Try to show the change on the sample review
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].str.lower()
```

6.2 remove the HTML and URLs from the reviews

[13]:
```
data_overall['review_body'] = data_overall['review_body'].replace(r'http\S+',␣
↪r' ', regex=True)

#Found that we have html tags
data_overall['review_body'] = data_overall['review_body'].replace(r'<.*?>', r'␣
↪', regex = True)
```

```
#Try to show the change on sample review
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].
 ↪replace(r'http\S+', r' ', regex=True)
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].replace(r'<.
 ↪*?>', r' ', regex = True)
```

6.3 perform contractions on the reviews.

[14]:
```
#Since i skip the apostrophe in the non-alphabetical part. I want to do this␣
 ↪part before non-alphabetical characters

def contractionfunction(s):
    s = contractions.fix(str(s))
    return s

data_overall['review_body'] = data_overall['review_body'].
 ↪apply(contractionfunction)


#Try to show the change on sample review
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].
 ↪apply(contractionfunction)
```

6.4 remove non-alphabetical characters

[15]:
```
#Based on Professor's comment, non-alphabetical means numbers, signs, and␣
 ↪non-English charactors

data_overall['review_body'] = data_overall['review_body'].replace(r'[^a-zA-Z␣
 ↪]\s?',r' ',regex=True)


#Try to show the change on sample review
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].
 ↪replace(r'[^a-zA-Z ]\s?',r' ',regex=True)
```

6.5 Remove the extra spaces between the words

[16]:
```
data_overall['review_body'] = data_overall['review_body'].apply(lambda x: re.
 ↪sub(' +', ' ', x) )

#Try to show the change on sample review
```

```
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].
 ↪apply(lambda x: re.sub(' +', ' ', x) )
```

7 After Data Cleaning, Information needed to be printed

[17]:
```
avg_reviewlenafter = data_overall['review_body'].apply(len).mean()

print("Average length of reviews after data cleaning is "+␣
 ↪str(avg_reviewlenafter))
```

```
Average length of reviews after data cleaning is 310.06299
```

8 Before Data Pre-Processing, Information needed to be printed

[18]:
```
print("Average length of reviews before data pre-processing is "+␣
 ↪str(avg_reviewlenafter))


print("Three sample reviews after data cleaning")
print("First Sample")
print(sample3_review_bp.iloc[0,1])


print("Second Sample")
print(sample3_review_bp.iloc[1,1])


print("Third Sample")
print(sample3_review_bp.iloc[2,1])
```

```
Average length of reviews before data pre-processing is 310.06299
Three sample reviews after data cleaning
First Sample
these gaskets are way too thin for the magic bullet the liquids splashes out of
the cup i tried placing two gaskets to see if it would seal but it does not work
do not waste your money on these gasket they do not fit the magic bullet
Second Sample
this toaster oven worked well while it lasted but it recently just suddenly
stopped working while it was in use and will not turn on again i cannot
recommend a toaster oven that only lasts for months of light use in a single
person household
Third Sample
i barely leave feedback on items i buy not because i do not want to but because
i cannot on the mobile app but this item pissed me off when i received it i
turned on the laptop just to give my irritated cents i received the mug in and i
was irritated and disappointed on the condition the mug was in it appeared to
```

have been broken or badly chipped and they glued the piece back on and repainted it you can clearly see the section that has a different shade of paint there is glue on the inside of the mug i cannot get off and i thought what company or person would honestly sell items like this in this condition i guess he is going to have to use it to hold pens at his office worthless

9 Pre-processing

9.1 remove the stop words

[19]:
```
from nltk.corpus import stopwords

#First we tokizen the word
#data_overall['review_body'] = data_overall['review_body'].apply(lambda x: nltk.
 ↪word_tokenize(x))


stop = stopwords.words('english')
#data_overall['review_body'] = data_overall['review_body'].apply(lambda x:␣
 ↪[word for word in x if word not in (stop)])

#print(data_overall['review_body'])


data_overall['review_body'] = data_overall['review_body'].apply(lambda x: ' '.
 ↪join([word for word in x.split() if word not in (stop)]))


#Test the changes in the sample we pick
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].
 ↪apply(lambda x: ' '.join([word for word in x.split() if word not in (stop)]))
```

9.2 perform lemmatization

[20]:
```
from nltk.stem import WordNetLemmatizer

lemmatizer = nltk.stem.WordNetLemmatizer()
#data_overall['review_body'] = data_overall['review_body'].apply(lambda x:␣
 ↪[lemmatizer.lemmatize(word) for word in x])

#print(data_overall['review_body'])


data_overall['review_body'] = data_overall['review_body'].apply(lambda x: ' '.
 ↪join([lemmatizer.lemmatize(word) for word in x.split()]))
```

```
#Test the changes in the sample we pick
sample3_review_bp['review_body'] = sample3_review_bp['review_body'].
 ↪apply(lambda x: ' '.join([lemmatizer.lemmatize(word) for word in x.split()]))
```

10 After Data Pre-Processing, Information needed to be printed

[21]:
```
avg_reviewlenafterdp = data_overall['review_body'].apply(len).mean()

print("Average length of reviews after data pre-processing is "+␣
 ↪str(avg_reviewlenafterdp))

print("Three sample reviews after data pre-processing")
print("First Sample")
print(sample3_review_bp.iloc[0,1])


print("Second Sample")
print(sample3_review_bp.iloc[1,1])


print("Third Sample")
print(sample3_review_bp.iloc[2,1])
```

```
Average length of reviews after data pre-processing is 189.93728
Three sample reviews after data pre-processing
First Sample
gasket way thin magic bullet liquid splash cup tried placing two gasket see
would seal work waste money gasket fit magic bullet
Second Sample
toaster oven worked well lasted recently suddenly stopped working use turn
cannot recommend toaster oven last month light use single person household
Third Sample
barely leave feedback item buy want cannot mobile app item pissed received
turned laptop give irritated cent received mug irritated disappointed condition
mug appeared broken badly chipped glued piece back repainted clearly see section
different shade paint glue inside mug cannot get thought company person would
honestly sell item like condition guess going use hold pen office worthless
```

11 TF-IDF Feature Extraction

[22]:
```
import sklearn

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

```
from sklearn.feature_extraction.text import TfidfVectorizer

#First split the data to train and testing
#20% test and 80% train and 20% test

y = data_overall['star_rating']
x = data_overall['review_body']

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


vectorizer = TfidfVectorizer()
x_train_tfidf = vectorizer.fit_transform(x_train)

#use matrix to test
x_test_tfidf = vectorizer.transform(x_test)
```

12 Perceptron

[23]:
```
from sklearn.linear_model import Perceptron
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import precision_score, recall_score, f1_score,␣
 ↪accuracy_score

#Train data
perception_model = Perceptron()
perception_model.fit(x_train_tfidf, y_train)

#Perdict train data
y_train_pred = perception_model.predict(x_train_tfidf)

#Compare Accuracy, Precision, Recall, and f1-score
accuracy_train = accuracy_score(y_train, y_train_pred)
precision_train = precision_score(y_train, y_train_pred)
recall_train = recall_score(y_train, y_train_pred)
f1score_train = f1_score(y_train, y_train_pred)

print("Perceptron: Accuracy, Precision, Recall, F1-Score")
print("Train", accuracy_train, precision_train, recall_train, f1score_train,␣
 ↪sep = ", ")

#Predict the test

y_test_pred = perception_model.predict(x_test_tfidf)
```

```
#Compare Accuracy, Precision, Recall, and f1-score
accuracy_test = accuracy_score(y_test, y_test_pred)
precision_test = precision_score(y_test, y_test_pred)
recall_test = recall_score(y_test, y_test_pred)
f1score_test = f1_score(y_test, y_test_pred)


print("Test", accuracy_test, precision_test, recall_test, f1score_test, sep =␣
 ↪", ")
```

```
Perceptron: Accuracy, Precision, Recall, F1-Score
Train, 0.903575, 0.8977372109924492, 0.9109327933805792, 0.90428686643092
Test, 0.858325, 0.8552018248537142, 0.862601910668734, 0.858885928434473
```

13 SVM

[24]:
```
from sklearn.svm import LinearSVC

svm_model = LinearSVC()
svm_model.fit(x_train_tfidf, y_train)

#Perdict train data
y_trainsvm_pred = svm_model.predict(x_train_tfidf)

#Compare Accuracy, Precision, Recall, and f1-score
accuracysvm_train = accuracy_score(y_train, y_trainsvm_pred)
precisionsvm_train = precision_score(y_train, y_trainsvm_pred)
recallsvm_train = recall_score(y_train, y_trainsvm_pred)
f1scoresvm_train = f1_score(y_train, y_trainsvm_pred)


print("SVM,: Accuracy, Precision, Recall, F1-Score")
print("Train", accuracysvm_train, precisionsvm_train, recallsvm_train,␣
 ↪f1scoresvm_train, sep = ", ")

#Predict test data
y_testsvm_pred = svm_model.predict(x_test_tfidf)


#Compare Accuracy, Precision, Recall, and f1-score
accuracysvm_test = accuracy_score(y_test, y_testsvm_pred)
precisionsvm_test = precision_score(y_test, y_testsvm_pred)
recallsvm_test = recall_score(y_test, y_testsvm_pred)
f1scoresvm_test = f1_score(y_test, y_testsvm_pred)

print("Test", accuracysvm_test, precisionsvm_test, recallsvm_test,␣
 ↪f1scoresvm_test, sep = ", ")
```

```
SVM,: Accuracy, Precision, Recall, F1-Score
Train, 0.931125, 0.9321356535247617, 0.9299686277450723, 0.9310508796956728
Test, 0.8947, 0.8980878865849352, 0.8903616265692993, 0.8942080675139398
```

14 Logistic Regression

[25]:
```
from sklearn.linear_model import LogisticRegression

logistic_model = LogisticRegression(random_state=0, max_iter= 10000)
logistic_model.fit(x_train_tfidf, y_train)

#Perdict train data
y_trainlogistic_pred = logistic_model.predict(x_train_tfidf)
```

```
#Compare Accuracy, Precision, Recall, and f1-score
accuracylogistic_train = accuracy_score(y_train, y_trainlogistic_pred)
precisionlogistic_train = precision_score(y_train, y_trainlogistic_pred)
recalllogistic_train = recall_score(y_train, y_trainlogistic_pred)
f1scorelogistic_train = f1_score(y_train, y_trainlogistic_pred)


print("Logistic Regression: Accuracy, Precision, Recall, F1-Score")
print("Train", accuracylogistic_train, precisionlogistic_train,␣
 ↪recalllogistic_train, f1scorelogistic_train, sep = ", ")


#Predict test data
y_testlogistic_pred = logistic_model.predict(x_test_tfidf)


#Compare Accuracy, Precision, Recall, and f1-score
accuracylogistic_test = accuracy_score(y_test, y_testlogistic_pred)
precisionlogistic_test = precision_score(y_test, y_testlogistic_pred)
recalllogistic_test = recall_score(y_test, y_testlogistic_pred)
f1scorelogistic_test = f1_score(y_test, y_testlogistic_pred)

print("Test", accuracylogistic_test, precisionlogistic_test,␣
 ↪recalllogistic_test, f1scorelogistic_test, sep = ", ")
```

```
Logistic Regression: Accuracy, Precision, Recall, F1-Score
Train, 0.91253125, 0.9155775477827303, 0.9088829727398853, 0.9122179779086617
Test, 0.897675, 0.9029805352798054, 0.8910118541489521, 0.8969562699831323
```

15 Naive Bayes

[26]:
```
from sklearn.naive_bayes import MultinomialNB

bayes_model = MultinomialNB()
bayes_model.fit(x_train_tfidf, y_train)

#Perdict train data
y_trainbayes_pred = bayes_model.predict(x_train_tfidf)

#Compare Accuracy, Precision, Recall, and f1-score
accuracybayes_train = accuracy_score(y_train, y_trainbayes_pred)
precisionbayes_train = precision_score(y_train, y_trainbayes_pred)
recallbayes_train = recall_score(y_train, y_trainbayes_pred)
f1scorebayes_train = f1_score(y_train, y_trainbayes_pred)

print("Naive Bayes: Accuracy, Precision, Recall, F1-Score")
```

```
print("Train", accuracybayes_train, precisionbayes_train, recallbayes_train,␣
 ↪f1scorebayes_train, sep = ", ")

#Perdict test data
y_testbayes_pred = bayes_model.predict(x_test_tfidf)

#Compare Accuracy, Precision, Recall, and f1-score
accuracybayes_test = accuracy_score(y_test, y_testbayes_pred)
precisionbayes_test = precision_score(y_test, y_testbayes_pred)
recallbayes_test = recall_score(y_test, y_testbayes_pred)
f1scorebayes_test = f1_score(y_test, y_testbayes_pred)

print("Test", accuracybayes_test, precisionbayes_test, recallbayes_test,␣
 ↪f1scorebayes_test, sep = ", ")
```

```
Naive Bayes: Accuracy, Precision, Recall, F1-Score
Train, 0.8856, 0.8884649768946977, 0.8819353306585673, 0.8851881123530666
Test, 0.87165, 0.8794239313620347, 0.8613014555094283, 0.8702683580128368
```

[ ]: