Yelp Dataset Sentiment Analysis

Dataset - https://www.yelp.com/dataset/download (https://www.yelp.com/dataset/download)

Source code - https://github.com/iwiszhou/ML1010-final-project (https://github.com/iwiszhou/ML1010-final-project)

Group 10 - Haofeng Zhou - zhf85@my.yorku.ca (mailto:zhf85@my.yorku.ca)

This is a Yelp data-set. I would use this data-set to do a sentiment analysis. I would build a model to predict the review either positive or negative. This is a big data-set. Firstly, I would try to extra the review data and create a simple data-set, which only contain Review & Rating. After that, I would create a new column which is Class. Class column is either Positive or Negative. If Rating is grater than 3, I would mark Class to Positive. Otherwise, Negative. If I have more time at the end, I would introduce one more value to Class column which is Neutral (when Rating is equal to 3)

In [1]:
```
# Import libraries
import pandas as pd
import json
import numpy as np
import re
import nltk
import sqlite3
import matplotlib.pyplot as plt
from pathlib import Path
import os
import spacy
```

In [2]:
```
# Download stopwords if not existing
nltk.download('stopwords')
```

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

Out[2]:
```
True
```

In [3]:
```
# Set col to max width
pd.set_option('display.max_colwidth', -1)
```

In [4]:
```
# Database name & tables' name
db_name = "yelp.db"
table_names = {
    "reviews": "reviews",
    "clean_reviews": "clean_reviews"
}
```

In [18]:
```
# Helper functions

def get_absolute_path(file_name):
    return os.path.abspath('') + "/" + file_name


# Save data to database
def save_to_db(dataFrame, tableName):
    con = sqlite3.connect(db_name)
    dataFrame.to_sql(tableName, con)
    con.close()


# Get data (dataframe format) from database by table name
def get_table_by_name(tableName):
    con = sqlite3.connect(db_name)
    df = pd.read_sql_query("SELECT * FROM " + tableName + ";", con)
    con.close()
    return df


# Read data from file
# NOTE - the data-set is too big. I have already to several time, my comput
# 10000 rows. I would increase the data-set size when training the model.
def load_json():
    filename = get_absolute_path('./yelp_dataset/review.json')
    row_count = 0
    row_limit = 10000
    df = []
    with open(filename, encoding="utf8") as f:
        for line in f:
            df.append(json.loads(line))
            row_count = row_count + 1
            if row_count > row_limit:
                break
    df = pd.DataFrame(df)
    return df
```

STEP 1 - Gather data

In [20]:
```
# Get data from database or json file

file_path = get_absolute_path("data.csv")

if os.path.isfile(file_path):
    # Import csv
    df = pd.read_csv(file_path, encoding='utf-8')
else:
    df = load_json()
    # Export to csv
    df.to_csv(file_path, encoding='utf-8', index=False)


# Top 5 records
print(df.head().values)

# Shape of dataframe
print(df.shape)

# View data information
print(df.info())

# Check na values
print(df.isnull().values.sum())
```

```
[['Q1sbwvVQXV2734tPgoKj4Q' 'hG7b0MtEbXx5QzbzE6C_VA'
  'ujmEBvifdJM6h6RLv4wQIg' 1.0 6 1 0
  'Total bill for this horrible service? Over $8Gs. These crooks actually
had the nerve to charge us $69 for 3 pills. I checked online the pills ca
n be had for 19 cents EACH! Avoid Hospital ERs at all costs.'
  '2013-05-07 04:34:36']
 ['GJXCdrto3ASJOqKeVWPi6Q' 'yXQM5uF2jS6es16SJzNHfg'
  'NZnhc2sEQy3RmzKTZnqtwQ' 5.0 0 0 0
  "I *adore* Travis at the Hard Rock's new Kelly Cardenas Salon!  I'm alw
ays a fan of a great blowout and no stranger to the chains that offer thi
s service; however, Travis has taken the flawless blowout to a whole new
level!  \n\nTravis's greets you with his perfectly green swoosh in his ot
herwise perfectly styled black hair and a Vegas-worthy rockstar outfit.
Next comes the most relaxing and incredible shampoo -- where you get a fu
ll head message that could cure even the very worst migraine in minutes -
-- and the scented shampoo room.  Travis has freakishly strong fingers (i
n a good way) and use the perfect amount of pressure.  That was superb!
Then starts the glorious blowout... where not one, not two, but THREE peo
ple were involved in doing the best round-brush action my hair has ever s
een.  The team of stylists clearly gets along extremely well, as it's evi
dent from the way they talk to and help one another that it's really genu
ine and not some corporate requirement.  It was so much fun to be there!
\n\nNext Travis started with the flat iron.  The way he flipped his wrist
to get volume all around without over-doing it and making me look like a
Texas pagent girl was admirable.  It's also worth noting that he didn't f
ry my hair -- something that I've had happen before with less skilled sty
lists.  At the end of the blowout & style my hair was perfectly bouncey a
nd looked terrific.  The only thing better?  That this awesome blowout la
sted for days! \n\nTravis, I will see you every single time I'm out in Ve
gas.  You make me feel beauuuutiful!"
  '2017-01-14 21:30:33']
```

```
 ['2TzJjDVDEuAW6MR5Vuc1ug' 'n6-Gk65cPZL6Uz8qRm3NYw'
  'WTqjgwHlXbSFevF32_DJVw' 5.0 3 0 0
  "I have to say that this office really has it together, they are so org
anized and friendly!  Dr. J. Phillipp is a great dentist, very friendly a
nd professional.  The dental assistants that helped in my procedure were
amazing, Jewel and Bailey helped me to feel comfortable!  I don't have de
ntal insurance, but they have this insurance through their office you can
purchase for $80 something a year and this gave me 25% off all of my dent
al work, plus they helped me get signed up for care credit which I knew n
othing about before this visit!  I highly recommend this office for the n
ice synergy the whole office has!"
  '2016-11-09 20:09:03']
 ['yi0R0Ugj_xUx_Nek0-_Qig' 'dacAIZ6fTM6mqwW5uxkskg'
  'ikCg8xy5JIg_NGPx-MSIDA' 5.0 0 0 0
  "Went in for a lunch. Steak sandwich was delicious, and the Caesar sala
d had an absolutely delicious dressing, with a perfect amount of dressin
g, and distributed perfectly across each leaf. I know I'm going on about
the salad ... But it was perfect.\n\nDrink prices were pretty good.\n\nTh
e Server, Dawn, was friendly and accommodating. Very happy with her.\n\nI
n summation, a great pub experience. Would go again!"
  '2018-01-09 20:56:38']
 ['11a8sVPMUFtaC7_ABRkmtw' 'ssoyf2_x0EQMed6fgHeMyQ'
  'b1b1eb3uo-w561D0ZfCEiQ' 1.0 7 0 0
  'Today was my second out of three sessions I had paid for. Although my
first session went well, I could tell Meredith had a particular enjoyment
for her male clients over her female. However, I returned because she did
my teeth fine and I was pleased with the results. When I went in today, I
was in the whitening room with three other gentlemen. My appointment star
ted out well, although, being a person who is in the service industry, I
always attend to my female clientele first when a couple arrives. Unbothe
red by those signs, I waited my turn. She checked on me once after my ori
ginal 30 minute timer to ask if I was ok. She attended my boyfriend on nu
merous occasions, as well as the other men, and would exit the room witho
ut even asking me or looking to see if I had any irritation. Half way thr
ough, another woman had showed up who she was explaining the deals to in
the lobby. While she admits timers must be reset half way through the pro
cess, she reset my boyfriends, left, rest the gentleman furthest away fro
m me who had time to come in, redeem his deal, get set, and gave his time
r done, before me, then left, and at this point my time was at 10 minute
s. So, she should have reset it 5 minutes ago, according to her. While I
sat there patiently this whole time with major pain in my gums, i watched
the time until the lamp shut off. Not only had she reset two others, expl
ained deals to other guest, but she never once checked on my time. When m
y light turned off, I released the stance of my mouth to a more relaxed s
tate, assuming I was only getting a thirty minute session instead of the
usual 45, because she had yet to come in. At this point, the teeth formul
a was not only burning the gum she neglected for 25 minutes now, but it b
egan to burn my lips. I began squealing and slapping my chair trying to g
et her attention from the other room in a panic. I was in so much pain, t
hat by the time she entered the room I was already out of my chair. She f
inally then acknowledged me, and asked if she could put vitamin E on my g
um burn (pictured below). At this point, she has treated two other gums b
urns, while neglecting me, and I was so irritated that I had to suffer, a
ll I wanted was to leave. While I waited for my boyfriend, she kept haras
sing me about the issue. Saying, "well burns come with teeth whitening."
While I totally agree, and under justifiable circumstances would not be a
s irritate, it could have easily been avoid if she had checked on me even
```

```
a second time, so I could let her know. Not only did she never check on m
y physical health, she couldn\'t even take two seconds to reset the time
r, which she even admitted to me. Her accuse was that she was coming in t
o do it, but I had the light off for a solid two minutes before I couldn
\'t stand the pain. She admitted it should be reset every 15 minutes, whi
ch means for 25 minutes she did not bother to help me at all. Her guest i
n the lobby then proceeded to attack me as well, simply because I wanted
to leave after the way I was treated. I also expected a refund for not ge
tting a complete session today, due to the neglect, and the fact I won\'t
be returning for my last, she had failed to do that. She was even screami
ng from the door, and continued to until my boyfriend and I were down the
steps. I have never in my life been more appalled by a grown woman\'s beh
avior, who claims to be in the business for "10 years." Admit your wrong
s, but don\'t make your guest feel unwelcome because you can\'t do you jo
b properly.'
  '2018-01-30 23:07:38']]
(10001, 9)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10001 entries, 0 to 10000
Data columns (total 9 columns):
review_id      10001 non-null object
user_id        10001 non-null object
business_id    10001 non-null object
stars          10001 non-null float64
useful         10001 non-null int64
funny          10001 non-null int64
cool           10001 non-null int64
text           10001 non-null object
date           10001 non-null object
dtypes: float64(1), int64(3), object(5)
memory usage: 703.3+ KB
None
0
```

There are not NA value in this data-set. Next, let's create a new column to store our Class/Label value, which depennds on our 'stars' column, if 'stars' is great than 3, Class/Label is 1 - 'Positive'. Otherwise, it is 0 - 'Negative'

In [21]:
```
# Create a class(label) column
def get_class_label_value(row):
    if row["stars"] >= 3:
        return 1
    return 0


review_file_path = get_absolute_path("review.csv")

if not os.path.isfile(review_file_path):
    df["class"] = df.apply(get_class_label_value, axis=1)

    # Create new data frame
    filter_df = df[['class', 'text']]
    print(filter_df.head(1).values)
    print(filter_df.shape[0])
    print(filter_df.columns)

    # Export to csv
    filter_df.to_csv(review_file_path, encoding='utf-8', index=False)
else:
    # Import csv
    filter_df = pd.read_csv(review_file_path, encoding='utf-8')
```

STEP 2 - Clean data / Text pre-processing

First of all, let's balance the data

In [22]:
```
balance_review_file_path = get_absolute_path("balance_review.csv")

if not os.path.isfile(balance_review_file_path):
    # num of Positive record
    print(filter_df.loc[filter_df["class"] == 1].count())

    # num of Negative record
    print(filter_df.loc[filter_df["class"] == 0].count())

    # balance the data
    balance_data_count = 10
    n_df = filter_df.loc[filter_df["class"] == 0][:balance_data_count]
    # number of negative rows
    print("Number of negative should be 100. Actual is ", len(n_df.loc[n_df
    print("Number of positive should be 0. Actual is ", len(n_df.loc[n_df["

    p_df = filter_df.loc[filter_df["class"] == 1][:balance_data_count]
    # number of positive rows
    print("Number of positive should be 100. Actual is ", len(p_df.loc[p_df
    print("Number of negative should be 0. Actual is ", len(p_df.loc[p_df["

    # merge positive and negative together to become a balance data
    filter_df = n_df.append(p_df)

    filter_df.to_csv(balance_review_file_path, encoding='utf-8', index=Fals
else:
    # Import csv
    filter_df = pd.read_csv(balance_review_file_path, encoding='utf-8')
```

Secondly, we would use NLTK method to normalize our corpus.

In [23]:
```
# Text Normalization - using NLTK
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')


def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z0-9\s]', '', doc, re.I)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc


normalize_corpus = np.vectorize(normalize_document)

# filter_df["norm_text"] = normalize_corpus(filter_df["text"])
#
# # Check the result
# print(filter_df["norm_text"].describe())
# print(filter_df.head(1))
```

As the result, the norm text is still have some words not fully converted to what we want. Such as, 'checked', 'costs', we expected those should stem correctly. Next, let's try library Spacy, which provide all lots of helper method for us to normalize our corpus.

Next, let's use Spacy to normolize text

In [24]:
```
nlp = spacy.load("en_core_web_sm")
white_list_pos = ["VERB", "PART", "NOUN", "ADJ", "ADV"]


def spacy_norm_text(text):
    # tokenizing
    doc = nlp(str(text))

    ret_set = set()

    # handle stop words, VERB, PART, ADJ, ADV and NOUN
    for token in doc:
        if not token.is_stop and token.text:  # remove stop words & empty s
            if token.pos_ in white_list_pos:  # if token is in white list,
                ret_set.add(token.lemma_.lower().strip())

    # handle PROPN
    for token in doc.ents:
        ret_set.add(token.text)

    # convert to list
    unique_list = list(ret_set)

    return " ".join(unique_list)


norm_review_file_path = get_absolute_path("norm_review.csv")

if not os.path.isfile(norm_review_file_path):
    filter_df["norm_text"] = filter_df.apply(lambda row: spacy_norm_text(rc

    # Export norm text to file
    filter_df.to_csv(norm_review_file_path, encoding='utf-8', index=False)
else:
    # Import norm text data frame
    filter_df = pd.read_csv(norm_review_file_path, encoding='utf-8')

# Check the result
print(filter_df["norm_text"].describe())
print(filter_df.head(1))
```

```
count     20
unique    20
top       love shabu perspective fresh home limited bland water taste pri
ce miserable try good be clean skip sauce favor well judge quality place
pot small hot selection expensive soup star appetite base quantity
freq      1
Name: norm_text, dtype: object
   class  \
0  0


text  \
0  Total bill for this horrible service? Over $8Gs. These crooks actually
had the nerve to charge us $69 for 3 pills. I checked online the pills ca
n be had for 19 cents EACH! Avoid Hospital ERs at all costs.
```

```
norm_text
0  pill service actually crook check total online charge 69 bill Avoid Ho
spital nerve avoid er 3 19 cents horrible cost hospital cent
```

STEP 3 - Feature extraction from text

Using TF-IDF to convert text to vector

In [25]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(min_df=2)
tfidf = vectorizer.fit_transform(filter_df["norm_text"].values)

# convert to array
tfidf = tfidf.toarray()
print(tfidf.shape)  # 200 is our rows, 1186 is how many words

words = vectorizer.get_feature_names()

# plt.figure(figsize=[20,4])
# _ = plt.show(tfidf)

pd.DataFrame(tfidf, columns=words)
```

```
(20, 222)
```

Out[25]:

	10	100	15	25	30	45	80	about	actually	
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.315025	0.00
1	0.174245	0.000000	0.095106	0.095106	0.087123	0.105399	0.000000	0.000000	0.000000	0.00
2	0.000000	0.000000	0.230908	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
3	0.000000	0.171843	0.155061	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.21
5	0.272400	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.164772	0.000000	0.00
6	0.123956	0.000000	0.000000	0.000000	0.123956	0.000000	0.000000	0.000000	0.149959	0.00
7	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
8	0.000000	0.119747	0.000000	0.000000	0.197965	0.119747	0.000000	0.000000	0.000000	0.00
9	0.000000	0.000000	0.000000	0.000000	0.146683	0.000000	0.000000	0.177453	0.000000	0.16
10	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
11	0.000000	0.000000	0.000000	0.227598	0.000000	0.000000	0.252229	0.000000	0.000000	0.00
12	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
13	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
14	0.000000	0.000000	0.000000	0.135768	0.000000	0.000000	0.000000	0.000000	0.000000	0.13
15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.207825	0.000000	0.000000	0.00
17	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
18	0.179586	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
19	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00

20 rows × 222 columns

STEP 4 - Build Models

In [26]:
```
from sklearn.model_selection import train_test_split

X = tfidf  # the features we want to analyze
y = filter_df['class'].values  # the labels, or answers, we want to test ag

# split into train and test dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

# Logistic regression
from sklearn.linear_model import LogisticRegression

model = LogisticRegression().fit(X_train, y_train)

predict_ret = model.predict_proba(X_test)

# convert to Positive and Negative
y_predict = np.array([int(p[1] > 0.5) for p in predict_ret])

# accuracy
print(y_predict)
print(y_test)
print(np.sum(y_test == y_predict) / len(y_test))
```

```
[0 1 1 1 0 0]
[0 0 1 1 0 1]
0.6666666666666666

/Users/iwiszhou/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_
model/logistic.py:432: FutureWarning: Default solver will be changed to
'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

In [ ]:
```
```