Student Information

Name: 謝安亭

Student ID: 110065516

GitHub ID: antinghsieh555

Instructions

1. First: do the **take home** exercises in the DM2022-Lab1-Master. You may need to copy some cells from the Lab notebook to this notebook. **This part is worth 20% of your grade.**

1. Second: follow the same process from the DM2022-Lab1-Master on **the new dataset**. You don't need to explain all details as we did (some **minimal comments** explaining your code are useful though). **This part is worth 30% of your grade.**
    - Download the the new dataset. The dataset contains a `sentence` and `score` label. Read the specificiations of the dataset for details.
    - You are allowed to use and modify the `helper` functions in the folder of the first lab session (notice they may need modification) or create your own.

1. Third: please attempt the following tasks on **the new dataset**. **This part is worth 30% of your grade.**
    - Generate meaningful **new data visualizations**. Refer to online resources and the Data Mining textbook for inspiration and ideas.
    - Generate **TF-IDF features** from the tokens of each text. This will generating a document matrix, however, the weights will be computed differently (using the TF-IDF value of each word per document as opposed to the word frequency). Refer to this Sciki-learn guide .
    - Implement a simple **Naive Bayes classifier** that automatically classifies the records into their categories. Use both the TF-IDF features and word frequency features to build two seperate classifiers. Comment on the differences. Refer to this article.

1. Fourth: In the lab, we applied each step really quickly just to illustrate how to work with your dataset. There are somethings that are not ideal or the most efficient/meaningful. Each dataset can be habdled differently as well. What are those inefficent parts you noticed? How can you improve the Data preprocessing for these specific datasets? **This part is worth 10% of your grade.**

1. Fifth: It's hard for us to follow if your code is messy, so please **tidy up your notebook** and **add minimal comments where needed**. **This part is worth 10% of your grade.**

You can submit your homework following these guidelines: Git Intro & How to hand your homework. Make sure to commit and save your changes to your repository **BEFORE the deadline (October 20th 11:59 pm, Thursday)**.

*** Section 3 ***

- Generate meaningful **new data visualizations**. Refer to online resources and the Data Mining textbook for inspiration and ideas.
- Generate **TF-IDF features** from the tokens of each text. This will generating a document matrix, however, the weights will be computed differently (using the TF-IDF value of each word per document as opposed to the word frequency). Refer to this Sciki-learn guide .
- Implement a simple **Naive Bayes classifier** that automatically classifies the records into their categories. Use both the TF-IDF features and word frequency features to build two seperate classifiers. Comment on the differences. Refer to this article.

Preprocessing

In [21]:
```
# import packages
import os
import pandas as pd
import helpers.data_mining_helpers as dmh

import matplotlib.pyplot as plt

from sklearn import preprocessing, metrics, decomposition, pipeline, dummy

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import metrics
import nltk
from nltk.corpus import stopwords
```

In [22]:
```
# get file
root = './sentiment labelled sentences/'

files = os.listdir(root)

if '.DS_Store' and 'readme.txt' in files:
    files.remove('.DS_Store')
    files.remove('readme.txt')

# get company name
companies = []
for file in files:
    companies.append(file.split("_")[0])

# read file
df_amazon = pd.read_csv(root+files[0], names=['text', 'score'], header=None,delimiter=
df_yelp = pd.read_csv(root+files[1], names=['text', 'score'], header=None,delimiter='\
df_imdb = pd.read_csv(root+files[2], names=['text', 'score'], header=None,delimiter='\
root = './sentiment labelled sentences/'

files = os.listdir(root)

if '.DS_Store' and 'readme.txt' in files:
    files.remove('.DS_Store')
    files.remove('readme.txt')

# get company name
companies = []
for file in files:
    companies.append(file.split("_")[0])

# read file
df_amazon = pd.read_csv(root+files[0], names=['text', 'score'], header=None,delimiter=
df_yelp = pd.read_csv(root+files[1], names=['text', 'score'], header=None,delimiter='\
df_imdb = pd.read_csv(root+files[2], names=['text', 'score'], header=None,delimiter='\
```

In [23]:
```
# add company name in the dataframe
df_amazon['company'] = companies[0]
```

```
df_yelp['company'] = companies[1]
df_imdb['company'] = companies[2]
```

In [24]:
```
df = pd.concat([df_amazon,df_yelp,df_imdb])
df
```

Out[24]:

	text	score	company
0	So there is no way for me to plug it in here i...	0	amazon
1	Good case, Excellent value.	1	amazon
2	Great for the jawbone.	1	amazon
3	Tied to charger for conversations lasting more...	0	amazon
4	The mic is great.	1	amazon
...	...	...	...
743	I just got bored watching Jessice Lange take h...	0	imdb
744	Unfortunately, any virtue in this film's produ...	0	imdb
745	In a word, it is embarrassing.	0	imdb
746	Exceptionally bad!	0	imdb
747	All in all its an insult to one's intelligence...	0	imdb

2748 rows × 3 columns

In [25]:
```
# Make company Name into dummy code
df["companyLabel"] = pd.factorize(df["company"])[0]
df
```

Out[25]:

	text	score	company	companyLabel
0	So there is no way for me to plug it in here i...	0	amazon	0
1	Good case, Excellent value.	1	amazon	0
2	Great for the jawbone.	1	amazon	0
3	Tied to charger for conversations lasting more...	0	amazon	0
4	The mic is great.	1	amazon	0
...	...	...	...	...
743	I just got bored watching Jessice Lange take h...	0	imdb	2
744	Unfortunately, any virtue in this film's produ...	0	imdb	2
745	In a word, it is embarrassing.	0	imdb	2
746	Exceptionally bad!	0	imdb	2
747	All in all its an insult to one's intelligence...	0	imdb	2

2748 rows × 4 columns

In [26]:
```
df[["company", "companyLabel"]]
```

Out[26]:

	company	companyLabel
0	amazon	0
1	amazon	0
2	amazon	0
3	amazon	0
4	amazon	0
...	...	...
743	imdb	2
744	imdb	2
745	imdb	2
746	imdb	2
747	imdb	2

2748 rows × 2 columns

In [27]:

```
# Make df['companyLabel'] into binary code
mlb = preprocessing.LabelBinarizer()
mlb.fit(df.companyLabel)
df['bin_companyLable'] = mlb.transform(df['companyLabel']).tolist()
df['bin_companyName'] = mlb.transform(df['companyLabel']).tolist()
df
```

Out[27]:

	text	score	company	companyLabel	bin_companyLable	bin_companyName
0	So there is no way for me to plug it in here i...	0	amazon	0	[1, 0, 0]	[1, 0, 0]
1	Good case, Excellent value.	1	amazon	0	[1, 0, 0]	[1, 0, 0]
2	Great for the jawbone.	1	amazon	0	[1, 0, 0]	[1, 0, 0]
3	Tied to charger for conversations lasting more...	0	amazon	0	[1, 0, 0]	[1, 0, 0]
4	The mic is great.	1	amazon	0	[1, 0, 0]	[1, 0, 0]
...	...	...	...	...	...	...
743	I just got bored watching Jessice Lange take h...	0	imdb	2	[0, 0, 1]	[0, 0, 1]
744	Unfortunately, any virtue in this film's produ...	0	imdb	2	[0, 0, 1]	[0, 0, 1]
745	In a word, it is embarrassing.	0	imdb	2	[0, 0, 1]	[0, 0, 1]
746	Exceptionally bad!	0	imdb	2	[0, 0, 1]	[0, 0, 1]
747	All in all its an insult to one's intelligence...	0	imdb	2	[0, 0, 1]	[0, 0, 1]

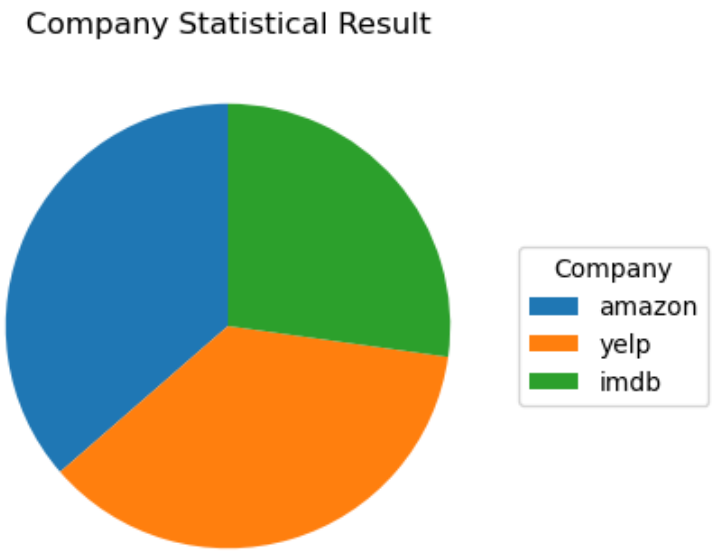
2748 rows × 6 columns

---

1. Generate meaningful new data visualizations. Refer to online resources and the Data Mining textbook for inspiration and ideas.

A. Pie chart to visualize company

In [28]:
```
# Pie chart
sta_data = df.company.value_counts()
# print (sta_data)

company_list = df.company.unique()
# print(company_list)

fig, ax = plt.subplots(figsize=(6,4), subplot_kw=dict(aspect="equal"))



wedges, texts = ax.pie(sta_data,
                       textprops=dict(color="w"),
                       startangle=90,)

ax.legend(wedges, company_list,
          title="Company",
          loc="center left",
          bbox_to_anchor=(1, 0, 0.5, 1))


ax.set_title("Company Statistical Result")

plt.show()
```



B. Word Cloud visualize text

In [29]:
```
# install packge
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /Users/antinghsieh/opt/anaconda3/envs/data
Mining/lib/python3.9/site-packages (1.8.2.2)
Requirement already satisfied: pillow in /Users/antinghsieh/opt/anaconda3/envs/dataMin
ing/lib/python3.9/site-packages (from wordcloud) (9.2.0)
Requirement already satisfied: matplotlib in /Users/antinghsieh/opt/anaconda3/envs/dat
aMining/lib/python3.9/site-packages (from wordcloud) (3.5.2)
Requirement already satisfied: numpy>=1.6.1 in /Users/antinghsieh/opt/anaconda3/envs/d
ataMining/lib/python3.9/site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: python-dateutil>=2.7 in /Users/antinghsieh/opt/anaconda
3/envs/dataMining/lib/python3.9/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /Users/antinghsieh/opt/anaconda3/envs/d
ataMining/lib/python3.9/site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/antinghsieh/opt/anaconda3/en
vs/dataMining/lib/python3.9/site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/antinghsieh/opt/anaconda3/e
nvs/dataMining/lib/python3.9/site-packages (from matplotlib->wordcloud) (1.4.2)
Requirement already satisfied: packaging>=20.0 in /Users/antinghsieh/opt/anaconda3/env
s/dataMining/lib/python3.9/site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in /Users/antinghsieh/opt/anaconda3/e
nvs/dataMining/lib/python3.9/site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: six>=1.5 in /Users/antinghsieh/opt/anaconda3/envs/dataM
ining/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotlib->wordcloud)
(1.16.0)
```

In [30]:
```
from wordcloud import WordCloud
from wordcloud import ImageColorGenerator
from wordcloud import STOPWORDS
```

- All comments word cloud

In [31]:
```
# draw word cloud figure
text = " ".join(i for i in df.text)
# text = " ".join(i for i in df.text)

stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, background_color="white").generate(text)
plt.figure(figsize=(8,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of All Comments')
plt.axis("off")
plt.show()
```

In [32]:
```
positive = df.loc[lambda f: f.score == '1'].text
```

In [33]:
```
negative = df.loc[lambda f: f.score == '0'].text
```

- Positive comments word cloud

In [34]:
```
# draw word cloud figure
text = " ".join(i for i in positive)
# text = " ".join(i for i in df.text)

stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords, colormap="Set2", background_color="white").
plt.figure(figsize=(8,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Positive Comments')
plt.axis("off")
plt.show()
```

- Negative comments word cloud

In [35]:
```
# draw word cloud figure
text = " ".join(i for i in negative)
# text = " ".join(i for i in df.text)

stopwords = set(STOPWORDS)
wordcloud = WordCloud(stopwords=stopwords,colormap="tab20c", background_color="white")
plt.figure(figsize=(8,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title('Word Cloud of Negative Comments')
plt.axis("off")
plt.show()
```

---

2. Generate TF-IDF features from the tokens of each text. This will generating a document matrix, however, the weights will be computed differently (using the TF-IDF value of each word per document as opposed to the word frequency). Refer to this Sciki-learn guide.

In [53]:
```
# Evaluation Matrice
def Evaluation(y_test, prediction):
    print("Confusion Matrix:")
    print(metrics.confusion_matrix(y_test, prediction))
    print("Accuracy: " , (metrics.accuracy_score(y_test, prediction)))
    print("Precision: " , (metrics.precision_score(y_test, prediction, pos_label = '1'
    print("Recall: " , (metrics.recall_score(y_test, prediction, pos_label = '1')))
    print("F-measure: " , (metrics.f1_score(y_test, prediction, pos_label = '1')))
```

- TF-IDF

In [38]:
```
# Get data
comment = df.text
comment
```

Out[38]:
```
0      So there is no way for me to plug it in here i...
1                            Good case, Excellent value.
2                                 Great for the jawbone.
3      Tied to charger for conversations lasting more...
4                                   The mic is great.
                             ...
743    I just got bored watching Jessice Lange take h...
744    Unfortunately, any virtue in this film's produ...
745                      In a word, it is embarrassing.
746                                Exceptionally bad!
747    All in all its an insult to one's intelligence...
Name: text, Length: 2748, dtype: object
```

In [40]:
```
vectorizer = TfidfVectorizer(smooth_idf=True)
tfidf = vectorizer.fit_transform(comment)
result = pd.DataFrame(tfidf.toarray(), columns=vectorizer.get_feature_names())
result
```

```
/Users/antinghsieh/opt/anaconda3/envs/dataMining/lib/python3.9/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[40]:

	00	10	100	11	12	13	15	15g	15pm	17	...	yucky	yukon	yum	yummy	yun	z500a
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2743	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2744	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2745	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2746	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0
2747	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0

2748 rows × 5155 columns

In [44]:
```
# Split the data into 75% train and 25% test data
Y_tfidf = df['score']
X_train, X_test, y_train, y_test= train_test_split(
    tfidf, Y_tfidf, test_size=0.25,random_state = 777)
```

In [52]:
```
# MultinomialNB
mnb1 = MultinomialNB()
mnb1.fit(X_train, y_train)
prediction = mnb1.predict(X_test)
Evaluation(y_test, prediction)
```

```
Confusion Matrix:
[[277  67]
 [ 54 289]]
Accuracy:  0.8238719068413392
Precision:  0.8117977528089888
Recall:  0.8425655976676385
F-measure:  0.8268955650929901
```

---

3. Implement a simple **Naive Bayes classifier** that automatically classifies the records into their categories. Use both the TF-IDF features and word frequency features to build two seperate classifiers. Comment on the differences. Refer to this article.

- Word frequency

In [54]:
```
count_vect = CountVectorizer()
X_counts = count_vect.fit_transform(df.text)
```

In [59]:
```
Y_counts = df['score']
X_train, X_test, y_train, y_test = train_test_split(
    X_counts, Y_counts, test_size=0.25,random_state = 777)
```

In [58]:
```
mnb2 = MultinomialNB()
mnb2.fit(X_train, y_train)
prediction = mnb2.predict(X_test)
Evaluation(y_test, prediction)
```

```
Confusion Matrix:
[[278  66]
 [ 66 277]]
Accuracy:  0.8078602620087336
Precision:  0.8075801749271136
Recall:  0.8075801749271136
F-measure:  0.8075801749271136
```

Comment on the differences. Refer to this article.

TfidfVectorizer and CountVectorizer both are methods for converting text data into vectors as model can process only numerical data. TF-IDF is a statistic that is based on the frequency of a word in the corpus but it also provides a numerical representation of how important a word is for statistical analysis. Count Vectorizer is a way to convert a given set of strings into a frequency representation.

From the above quantitative results, the score of TF-IDF is better than word frequency. This is due to TF-IDF not only focuses on the frequency of words present in the corpus but also provides the importance of the words.

*** Section 4 ***

In the lab, we applied each step really quickly just to illustrate how to work with your dataset. There are somethings that are not ideal or the most efficient/meaningful. Each dataset can be habdled differently as well. What are those inefficent parts you noticed? How can you improve the Data preprocessing for these specific datasets? This part is worth 10% of your grade.

- In Data preprocessing, we should also delete punctuation marks or connecting words such as and, or, also, etc. These words don't relate our classfication result, but they have negatve influence. Therefore, I have deleted stopwords in Part 3.
- In addition, when doing classification, we should also delete outliers to make the data more concentrated.