TARUMT
TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY

BMCS2013 Data Engineering

ASSIGNMENT 202301

Assignment Title	:	Sentiment Analysis of Domino's Pizza Malaysia
Programme	:	RDS2
Tutorial Group	:	G2

Declaration

I confirm that I have read and complied with all the terms and conditions of Tunku Abdul Rahman University College's plagiarism policy.

I declare that this assignment is free from all forms of plagiarism and for all intents and purposes is my own properly derived work.

Student Name	Student ID	% Contribution	Signature	Date
Sit Yie Sian	21WMR03693	30		1/5/23
Leong Sheng Mou	21WMR07568	20		1/5/23
Tan Jacqueline	21WMR03259	25		1/5/23
Loke Tze Min	21WMR07394	25		1/5/23
		100%		

Table of Contents

1. Introduction **4**
2. Extract, Transform, Load (ETL) / Extract, Load, Transform (ELT) **8**
2.1. ETL/ELT Steps 8
2.2 Criteria and Techniques Applied 13
2.3 Data Storage 14
2.3.1 MongoDB 14
2.3.2 HBase 14
Schema 14
File format 14
3. Machine Learning by using VADER **16**
4. Data Visualization **18**
Figure 4.5 **25**
References **26**
Appendix A Work Distribution Among Members **27**

1. Introduction

Domino's Pizza Malaysia is the Malaysian franchise of the American pizza restaurant chain Domino's Pizza(Universiti Teknologi Mara, 2017). The company was founded in Malaysia in 1997 and has since become one of the leading pizza delivery chains in the country. Domino's Pizza Malaysia has over 240 locations across the country, employing thousands of people and serving millions of customers each year. The company is committed to using high-quality ingredients and providing customer service to ensure that each customer has a satisfying experience with their pizza. However, we found that some of the customers were unsatisfied with the service of Dominos Pizza Malaysia as the customers leave their reviews in the official Facebook Page. Through this project, we tried to carry out sentiment analysis to analyze the reviews and determine whether the sentiment expressed is positive, negative, or neutral.

In this project, the dataset is obtained from Domino Malaysia official Facebook page. The method of data scraping is using python code. The dataset consists of commentID, commentURL, commenterID, CommenterName, commenterText and commentTime. The dataset will be used to undergo sentiment analysis to see the satisfaction of the commenters.

Columns	Explanations
commentID	The ID of each comment
commentURL	The URL of each comment
commenterID	The commenter ID which also know as profile ID
commenterName	The name of the commenter shown in Facebook
commenterText	The comment posted on the Domino official page.
CommentTime	The comment time of the comment. In this project, only year

	of comments will be taken.

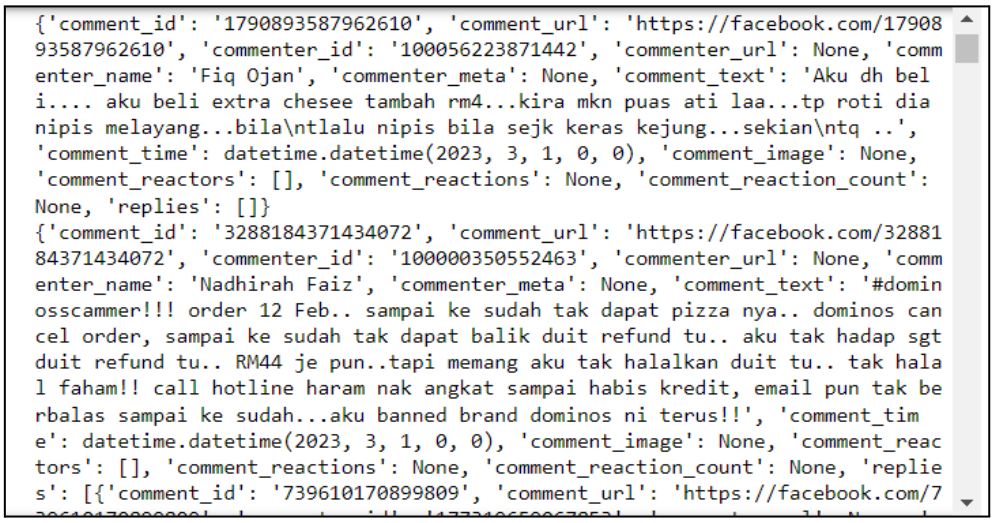
In the dataset scrapped, there are a total of 1044 sets of data being scrapped. As the purpose of the project is doing sentiment analysis, the irrelevant data has been removed such as commentID, commentURL, commenterID, commenterName. The data preprocessing is implemented to remove punctuation, remove null value. Tokenization and word normalization has been implemented as the dataset is a combination of English and Malay languages.

1.2 Data Source

As mentioned above, the dataset was scrapped manually with python code and stored into a CSV file. For the POST_ID, a third-party website, Comment Picker, was used to detect the post id for scraping purposes. The software used to do the scrapping code is Jupyter Notebook. After the data were scrapped, the data were copied and stored into a .csv file. The following shows the python code data scraping:

```python
import facebook_scraper as fs

# get POST_ID
POST_ID = "2781264325340027"

# number of comments to download
MAX_COMMENTS = True

# get the post
gen = fs.get_posts(
    post_urls=[POST_ID],
    options={"comments": MAX_COMMENTS, "progress": True}
)

# take 1st element of the generator which is the post we requested
post = next(gen)

# extract the comments part
comments = post['comments_full']

# process comments as you want
for comment in comments:

    # e.g. ...print them
    print(comment)

    # e.g. ...get the replies for them
    for reply in comment['replies']:
        print(' ', reply)
```

Figure 1.2.1 Python code for data scraping from Facebook



```
{'comment_id': '1790893587962610', 'comment_url': 'https://facebook.com/17908
93587962610', 'commenter_id': '100056223871442', 'commenter_url': None, 'comm
enter_name': 'Fiq Ojan', 'commenter_meta': None, 'comment_text': 'Aku dh bel
i.... aku beli extra chesee tambah rm4...kira mkn puas ati laa...tp roti dia
nipis melayang...bila\ntlalu nipis bila sejk keras kejung...sekian\ntq ..',
'comment_time': datetime.datetime(2023, 3, 1, 0, 0), 'comment_image': None,
'comment_reactors': [], 'comment_reactions': None, 'comment_reaction_count':
None, 'replies': []}
{'comment_id': '3288184371434072', 'comment_url': 'https://facebook.com/32881
84371434072', 'commenter_id': '100000350552463', 'commenter_url': None, 'comm
enter_name': 'Nadhirah Faiz', 'commenter_meta': None, 'comment_text': '#domin
osscammer!!! order 12 Feb.. sampai ke sudah tak dapat pizza nya.. dominos can
cel order, sampai ke sudah tak dapat balik duit refund tu.. aku tak hadap sgt
duit refund tu.. RM44 je pun..tapi memang aku tak halalkan duit tu.. tak hala
l faham!! call hotline haram nak angkat sampai habis kredit, email pun tak be
rbalas sampai ke sudah...aku banned brand dominos ni terus!!', 'comment_tim
e': datetime.datetime(2023, 3, 1, 0, 0), 'comment_image': None, 'comment_reac
tors': [], 'comment_reactions': None, 'comment_reaction_count': None, 'replie
s': [{'comment_id': '739610170899809', 'comment_url': 'https://facebook.com/7
```

Figure 1.2.2 Parts of output for data scrapped

```
from kafka import KafkaAdminClient, KafkaConsumer, KafkaProducer
from kafka.admin import NewTopic
import json
from json import loads
from csv import DictReader
```

Figure 1.2.3 Import necessary library

The following part is the setup for Kafka. Initially, the data were stored into csv file type. In this step, the data were converted into json type for upload to Kafka. The output shown are the acknowledgement of whether the data has been uploaded to the Kafka and the partition number were shown.

```
#setting for Kafka Producer
bootstrap_servers = ['localhost:9092']
topicname = 'domino.topic.raw'
producer = KafkaProducer(bootstrap_servers = bootstrap_servers)
producer = KafkaProducer()


# iterate each line as an ordered dictionary and print few column by column name
with open('Domino_Dataset.csv','r') as read_data:
    csv_dict_reader = DictReader(read_data)
    for row in csv_dict_reader:
        ack = producer.send(topicname,json.dumps(row).encode('utf-8'))
        metadata = ack.get()
        print(metadata.topic, metadata.partition)
```

```
domino.topic.raw 1
domino.topic.raw 0
domino.topic.raw 1
domino.topic.raw 1
domino.topic.raw 1
domino.topic.raw 1
domino.topic.raw 0
domino.topic.raw 1
domino.topic.raw 1
```

Figure 1.2.4 Setup for Kafka producer, convert file to json and upload to kafka.

2. Extract, Transform, Load (ETL) / Extract, Load, Transform (ELT)

2.1. ETL/ELT Steps

1. Extract: The first step of ETL/ELT is to extract data from a source system. In this code, the source system is Apache Kafka. The code uses the KafkaConsumer class to connect to the 'domino.topic.raw' topic in Kafka and consume the messages produced by the producer.

```
# the Kafka bootstrap server
hostname = socket.gethostname()
bootstrap_servers = f"{hostname}:9092"

# Set up the Kafka consumer
consumer = KafkaConsumer('domino.topic.raw',
                         bootstrap_servers=bootstrap_servers,
                         auto_offset_reset='earliest',
                         value_deserializer=lambda x:
json.loads(x.decode('utf-8')))
```

Figure 2.1.1 Set up Kafka Consumer

2. Transform: The next step of ETL/ELT is to transform the extracted data into a desired format or structure. In this code, the transformation is performed on the 'commenterText' column of the data. The following transformations are applied:

- Non-ASCII characters are removed from the text using a regular expression.
- Substrings starting with /u or /U (emoji in Facebook) are removed using another regular expression.
- Punctuation and numbers are removed from the text using another regular expression.
- The text is converted to lowercase using the lower() function.
- The text is tokenized into individual words using the word_tokenize() function from the NLTK library.

- The tokens are normalized using a dictionary of English and Malay language normalizers.
- The normalized tokens are joined back into a string.

```
combined_normalizer = {**english_normalizer, **malay_normalizer}
def preprocess_text(text):
    global normalized_tokens
    # Remove ASCII characters
    text = re.sub(r'[^\x00-\x7F]+', '', text)
    # Remove substrings starting with /u or /U (emoji in Facebook)
    text = re.sub(r'\/[uU]\S+', '', text)
    # Remove punctuations and numbers
    text = re.sub('[^a-zA-Z]', ' ', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenize the text
    tokens = word_tokenize(text)
    # Normalize tokens using the combined normalizer dictionary
    normalized_tokens = [combined_normalizer.get(token, token) for token in tokens]
    # Join the tokens back into a string
    preprocessed_text = ' '.join(normalized_tokens)
    return preprocessed_text
```

Figure 2.1.2 Preprocess the comment text

```
english_normalizer = {
    'tmr': 'tomorrow',
    'u': 'you',
    'r': 'are',
    'pls': 'please',
    'plz': 'please',
    'thx': 'thanks',
    'ty': 'thank you',
    'btw': 'by the way',
    'brb': 'be right back',
    'idk': 'i do not know',
    'imho': 'in my humble opinion',
    'lol': 'laugh out loud',
    'omg': 'oh my god',
    'tldr': 'too long did not read',
    'ttyl': 'talk to you later',
    'yw': 'you are welcome',
    'np': 'no problem',
```

Figure 2.1.3 Part of English normalizer

```
malay_normalizer = {
    'dpt': 'dapat',
    'knp': 'kenapa',
    'sy': 'saya',
    'dh': 'sudah',
    'yg': 'yang',
    'dgn': 'dengan',
    'dlm': 'dalam',
    'org': 'orang',
    'blm': 'belum',
    'jd': 'jadi',
    'bkn': 'bukan',
    'tlg': 'tolong',
    'pstu': 'pastu',
    'skrg': 'sekarang',
    'sm': 'sama',
    'tmpt': 'tempat',
    'bg': 'bagi',
```

Figure 2.1.4 Part of Malay normalizer

3. Load: The final step of ETL/ELT is to load the transformed data into a target system. In this code, the target system is Apache HBase. The code uses the HappyBase library to connect to an HBase instance and insert the transformed data into an HBase table named 'domino'.

```
# Connect to HBase
connection = happybase.Connection(port=9090)
```

Figure 2.1.5 Build a connection

```
table_name = 'domino'
# Check if the table exists
if table_name.encode() in connection.tables():
    # Disable the table before deleting
    connection.disable_table(table_name)
    # Delete the table
    connection.delete_table(table_name)
    print(f"Table {table_name} deleted successfully.")
else:
    print(f"Table {table_name} does not exist.")
```

Figure 2.1.6 Delete existing table

```
# Create the table
families = {'data': {}}
connection.create_table(table_name, families)
table = connection.table(table_name)
```

Figure 2.1.7 Create table

```
seq = 0
# Continuously poll for new messages
for message in consumer:
    # Extract the JSON data from the Kafka message
    json_data = message.value

    # Preprocess the 'commenterText' column
    commenter_text_clean = preprocess_text(json_data['commenterText'])

    # Extract year from 'commentTime' column
    comment_time = extract_year(json_data['commentTime'])

    # Write the data to HBase
    row_key = f"{seq}"
    data = {
        'data:commenterName': str(json_data['commenterName']),
        'data:commenterText': str(commenter_text_clean),
        'data:commentTime': str(comment_time)
    }
    # Convert data to bytes before uploading to HBase
    byte_data = {key.encode(): value.encode() for key, value in data.items()}
    table.put(row_key.encode(), byte_data)
    seq += 1

    # Print the current state of the HBase table
    print(f"HBase table '{table_name}': {len(list(table.scan()))} rows")
```

Figure 2.1.9 Clean the data then upload to Hbase

For each new message in the consumer, the code extracts the JSON data from the Kafka message and preprocesses the commenterText column using the preprocess_text function. It then extracts the year from the commentTime column using the extract_year function.

Finally, the code writes the transformed data to the domino table in HBase, using a unique row key generated from a sequence seq. The data is stored as a dictionary of column qualifiers and values, where the column family is data. The table.put method is used to insert the data into HBase.

After each new row is inserted, the code prints the current state of the domino table, including the number of rows. This allows us to monitor the progress of the ETL process.

2.2 MongoDB

After preprocessing, data can be stored in MongoDB as a collection of JSON-like documents, where each document represents a single comment. Each document contains fields such as _id, commentText, commentYear, sentiment_label, and Word Length. To retrieve data from MongoDB, one can use the find() method to query the collection based on specific criteria

2.3 HBase

Schema

The HBase table domino has a single column family named data. The column family contains the following columns:

- commenterName: A string column that stores the name of the commenter.
- commenterText: A string column that stores the preprocessed comment text of the commenter.
- commentTime: A string column that stores the year extracted from the comment time.

File format

HBase stores data in a column-oriented format. Each row in a table is identified by a unique row key, and each row can have multiple columns. Each column is identified by a column family and a column qualifier. In the provided code, we have only one column family named data, and each column in this column family is identified by its column qualifier.

Example

```
{
   "commenterName": "John Doe",
   "commenterText": "Hello world",
   "commentTime": "2023"
}
```

The provided code performs the following steps to store this data in the HBase table:

1. The comment text is preprocessed by removing non-ASCII characters, emoji, punctuation, and numbers, and then normalized using a dictionary of common abbreviations and slang terms.
2. The year is extracted from the comment time string.
3. The data is stored in the domino HBase table using a row key that increments for each new message received. The data is stored in the data column family with the commenterName, commenterText, and commentTime columns.

   So, after processing the above message, the domino table in HBase would contain the following row:

```
Row key: b'0'
b'data:commentTime': b'2023'
b'data:commenterName': b' BabySandra Kayrool Carolina Venus'
b'data:commenterText': b'yang set regular itu mantap dapat dua keping c
rumah sambil nonton netflix'
```

Figure 3.2.2 Hbase table

3.0 Machine Learning by using VADER

```
# Set up connection to MongoDB
client = MongoClient("mongodb://localhost:27017/")
db = client["domino_clean"]
collection = db["comments"]

# Load data from MongoDB collection into a pandas DataFrame
comments = list(collection.find())
df = pd.DataFrame(comments)
print(df.shape)
```

Retrieves data from a MongoDB database using PyMongo and loads it into a pandas DataFrame. Specifically, it connects to a MongoDB instance running on the localhost, selects a database called "domino1", and a collection called "domino_comments". It then retrieves all the documents in the collection and converts them into a list, which is used to create a pandas DataFrame called "df". This DataFrame contains all the comments in the "domino_comments" collection.

```
from pymongo import MongoClient
from textblob import TextBlob
import pandas as pd
from hdfs import InsecureClient
from nltk.sentiment import SentimentIntensityAnalyzer
```

```
# Define a function to perform sentiment analysis on each comment
def analyze_sentiment(comment):
    return TextBlob(comment).sentiment.polarity

# Apply the sentiment analysis function to each comment and reduce the sentiment scores
sentiment_scores = df["commenterText"].apply(analyze_sentiment)
df["sentiment_score"] = sentiment_scores
sentiment_df = df.groupby("commenterName").agg({
    "sentiment_score": "mean",
    "commenterText": lambda x: x.tolist(),
    "commentTime": lambda x: x.tolist()
}).reset_index()
sentiment_df = sentiment_df.explode("commentTime")
sentiment_df["sentiment_label"] = sentiment_df["sentiment_score"].apply(lambda score: "positive" if score > 0 else "negative" if score < 0 else "neutral")
sentiment_df = sentiment_df[["commenterName", "commentTime", "commenterText", "sentiment_score", "sentiment_label"]]
```

Sentiment analysis on comments using the technique in Python. TextBlob uses a rule-based approach and a machine learning approach for sentiment analysis. In this code, the sentiment analysis is performed using the rule-based approach of TextBlob, which computes a polarity score for each comment between -1 (negative) and 1 (positive). The sentiment score is then aggregated and transformed using PySpark's DataFrame APIs, and the resulting sentiment label is categorized as positive, negative, or neutral.

```
# Save the sentiment analysis results to a pandas DataFrame
sentiment_csv = sentiment_df.to_csv(index=False)

# Connect to HDFS and write the DataFrame to a file in HDFS
hdfs_client = InsecureClient("http://10.123.51.209:50070", user="user2")
with hdfs_client.write("/user/user2/data/domino/domino_sentiment_analysis.csv", encoding="utf-8", overwrite=True) as writer:
    writer.write(sentiment_csv)
```

Store domino sentiment analysis through Hadoop Distributed File System (HDFS) and write a DataFrame to a CSV file stored in HDFS. It creates an InsecureClient object to connect to the HDFS namenode at the specified URL with the specified user credentials. Then, it uses the write method of the hdfs_client object to write the CSV data to the specified HDFS file path. The with statement ensures that the file is closed properly after writing is complete. The encoding parameter specifies the character encoding used in the file, and overwrite=True ensures that any existing file with the same name is overwritten.

```
(LeongShengMou) [user2@vm122 ~]$ hdfs dfs -ls data/domino/
Found 1 items
-rw-r--r--   3 user2 hadoop  458142696 2023-05-01 12:44 data/domino/domino_sentiment_analysis.csv
```

4. Data Visualization

According to Figure 4.1, we have visualized the total number of each sentiment label. We found that 403 customers are neutral with Domino's Pizza which has the most number. 313 customers are satisfied with service and quality of the food of Domino's Pizza. However, 306 of customers are dissatisfied with the Domino's Pizza due to multiple reason.

Visualize total number of each sentiment label

```
[26]: plt.figure(figsize=(20,8))

x = sentiment_df["sentiment_label"].value_counts().sort_index().index
y = sentiment_df["sentiment_label"].value_counts().sort_index().values
for i in range(len(x)):
    height = y[i]
    plt.text(x[i], height + 0.5, '%.1f' %height, ha='center', va='bottom', size = 12)
plt.title("Number of sentiment label")
plt.xlabel("Sentiment Label")
plt.ylabel("Number of each sentiment")
plt.bar(x,y,color='#8fd0df')
```

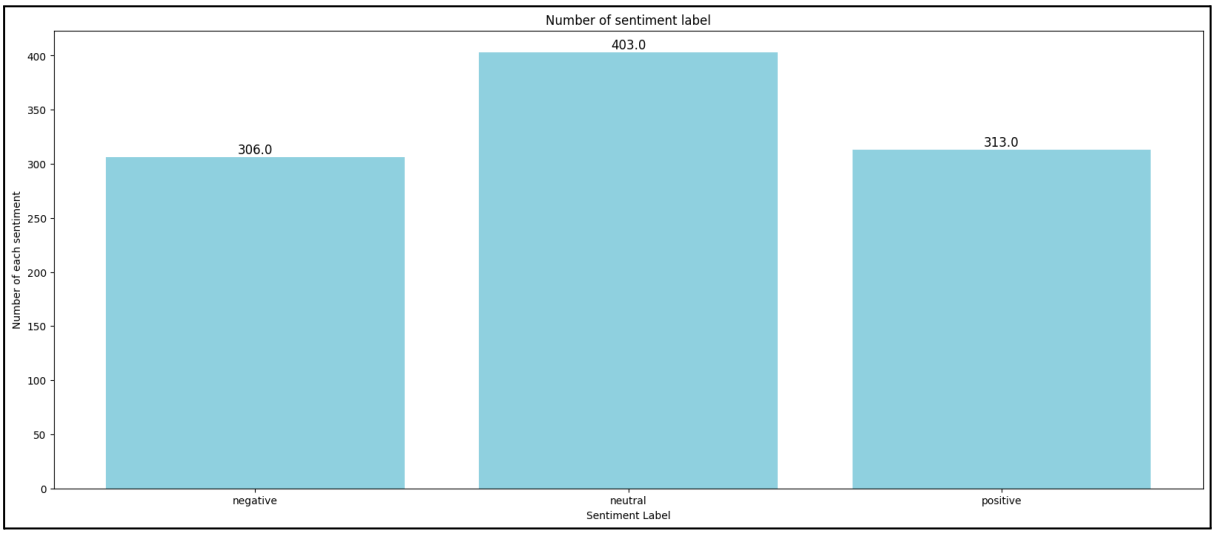


Figure 5.1

According to figure 5.2, the codes try to show the customer name who leaves the reviews. There are some of the customers that leave only one comment, but some customers leave more than one comment. We assume that those customers who leave more than one comment in the facebook page are the loyal customers from Domino's Pizza Malaysia because they had visited Domino's multiple times and they are able to leave more than one. Also, Domino's Pizza Malaysia customer service had replied to 25 more customers among 1022 rows.

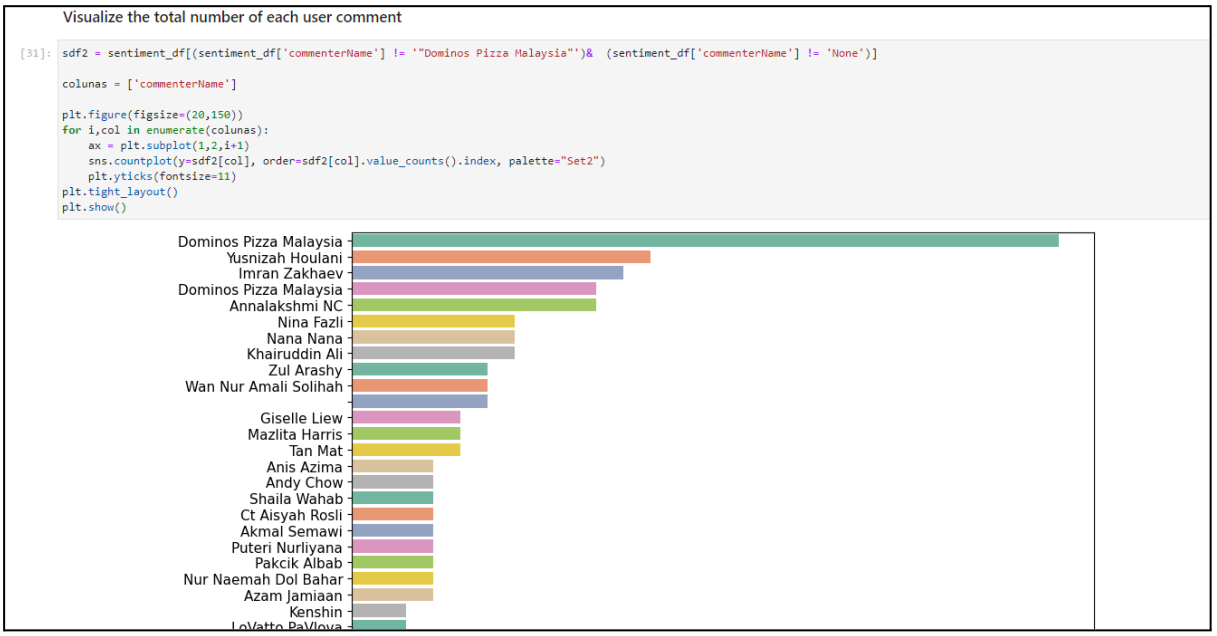


Figure 5.2

According to figure 4.3, the word count in each commentText sentence is calculated. We can see that the longer the word count the higher the negative or positive score. Customers who are dissatisfied may want to provide more detailed feedback on what went wrong or what they didn't like about their experience. This could lead to longer reviews as they try to provide as much information as possible. Sometimes people write longer reviews as a way to vent their frustration and disappointment about the experience. Writing can be a therapeutic outlet for people to express their emotions and feelings. This is similar to satisfied customers as they may want to share details about the excellent service they received, the high quality of the product, or other aspects that exceeded their expectations.

Visualize sentiment score

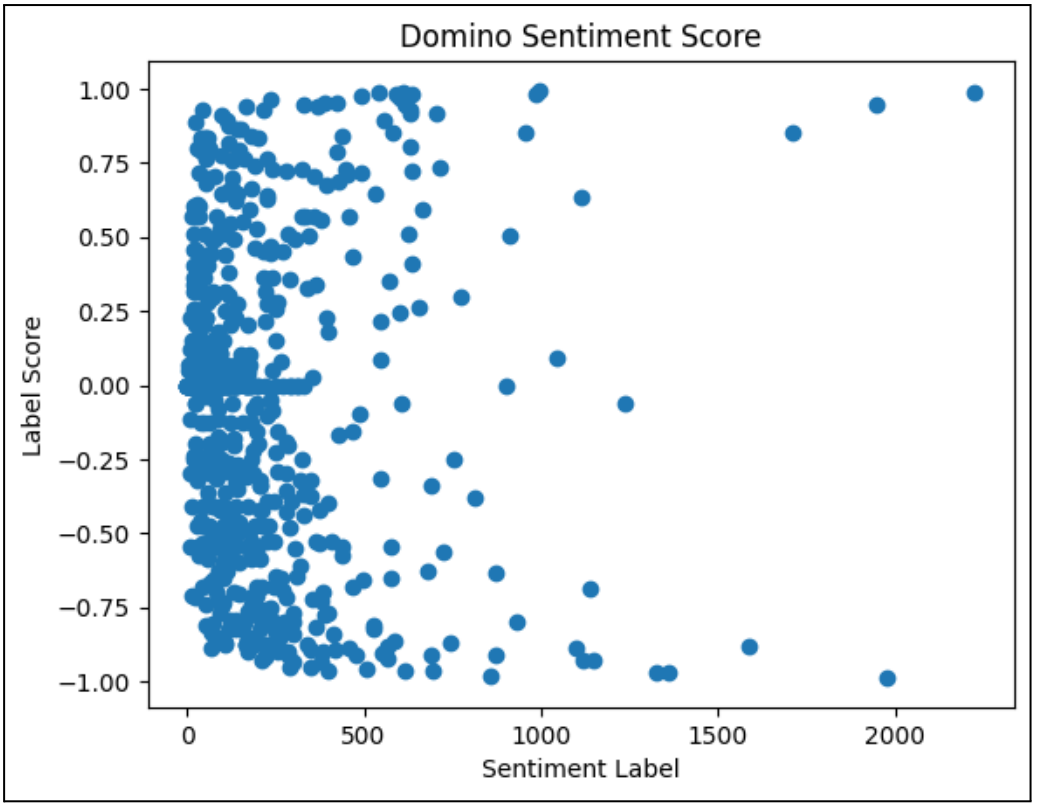
```
[25]: sentiment_df['Wordlength'] = sentiment_df['commenterText'].apply(lambda x: len(' '.join(x).split()))
      sentiment_df
```

[25]:

	commenterName	commentTime	commenterText	sentiment_score	sentiment_label	Wordlength
0	????? ???????	2023	[Wan Azizi Wan Amran]	-0.200000	negative	4
1	?????? ???	2023	[Fatihah Omar]	0.000000	neutral	2
2	Aidil Raof	2023	[hi..domino dah x buat cheese tarik ke dlm rot...	0.000000	neutral	10
3	Aimi Rusli	2023	[Dominoa dia dh match kan..ada air gas pulak??...	0.000000	neutral	8
4	Ainna Afiqah	2023	[Ainna Aqilah]	0.000000	neutral	2
...	...	...	...	...	...	...
856	Zul Arashy	2023	[Wan Nur Amali Solihah ??, Wan Nur Amali Solih...	-0.200000	negative	38
857	Zul Hz	2023	[19 tapi roti yg paling nipis dia bagi baru ak...	0.000000	neutral	12
858	Zulfitri Muhamad	2023	[2 hari lepas sy pergi domino's pizza kemaman....	0.500000	positive	68
859	Zulhazmi	14/1/2018	[Damansara Jelatek is the best ever outlet!\n\...	0.398907	positive	125
860	Zura Kamarul Hussein	21/8/2017	[Dominos putra perdana puchong 21 August 2017 ...	0.124852	positive	137

1044 rows × 6 columns

```
[27]: # create a figure and axis
      fig, ax = plt.subplots()

      # scatter the sepal_length against the sepal_width
      ax.scatter(sentiment_df['Wordlength'], sentiment_df['sentiment_score'])
      # set a title and labels
      ax.set_title('Domino Sentiment Score')
      ax.set_xlabel('Sentiment Label')
      ax.set_ylabel('Label Score')
```



[28]:
```
# create a figure and axis
fig, ax = plt.subplots()

# scatter the sepal_length against the sepal_width
ax.scatter(sentiment_df['Wordlength'], sentiment_df['sentiment_label'])
# set a title and labels
ax.set_title('Domino Sentiment Label')
ax.set_xlabel('Sentiment Label')
ax.set_ylabel('Label Score')
```

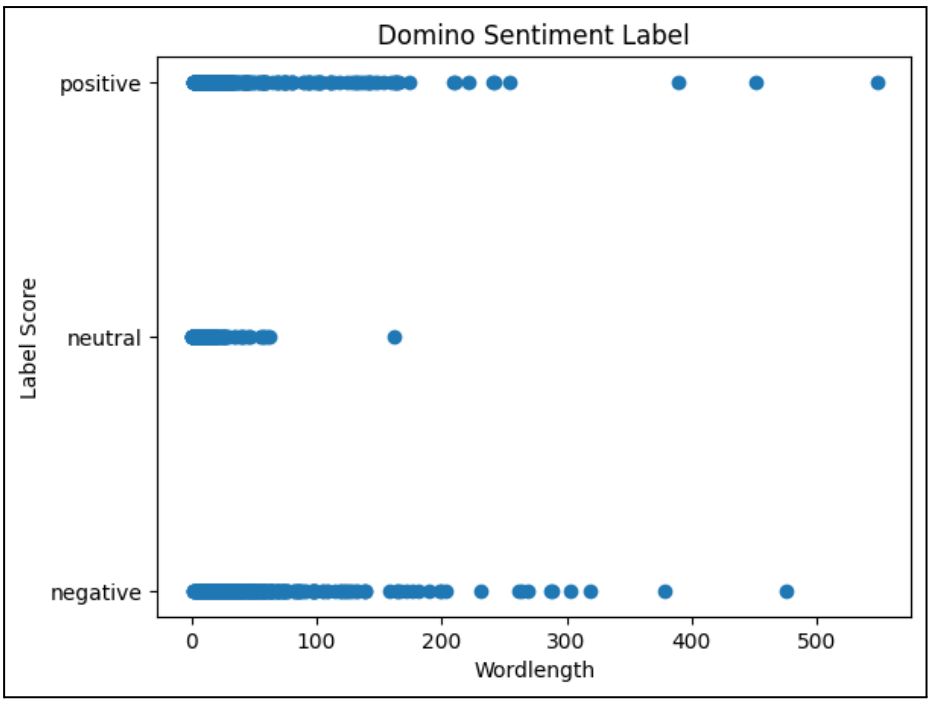


Figure 4.3

Figure 4.4 indicates the sentence length mean, min, max, and count for each sentiment label and a bar plot is plotted.

```
[39]: grouped = sentiment_df.groupby('sentiment_label')['Wordlength'].agg(['mean', 'min', 'max', 'count'])
      print(grouped)

                          mean  min   max  count
      sentiment_label
      negative        246.441176    7  1978    306
      neutral          41.754342    0   902    403
      positive        188.769968    5  2225    313

[48]: #Word cloud for positive, negative, neutral label
      # create a figure and axis

      grouped = sentiment_df.groupby('sentiment_label')['Wordlength'].agg(['mean', 'min', 'max', 'count'])
      ax = grouped.plot(kind='bar', figsize=(8, 6))

      # set the title and labels
      ax.set_title('Word Length of Comments by Sentiment Label')
      ax.set_xlabel('Sentiment Label')
      ax.set_ylabel('Word Length')
```

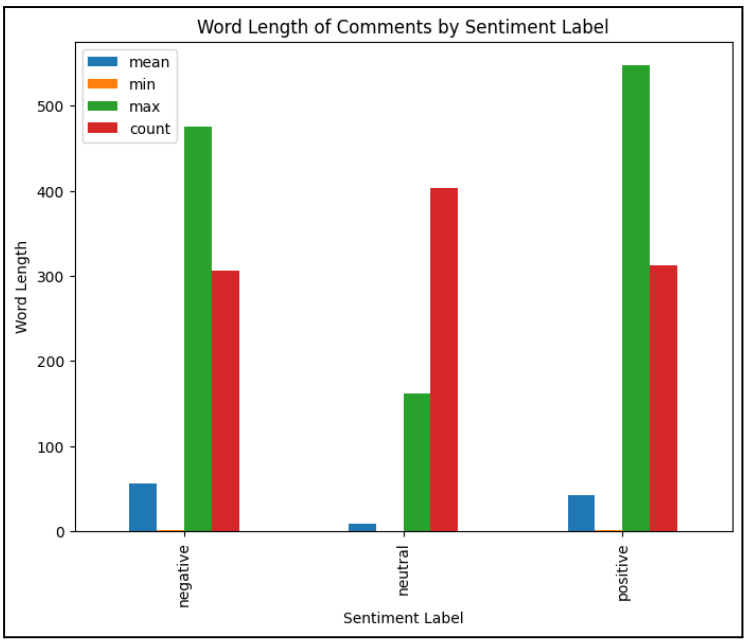

Word Length of Comments by Sentiment Label
mean
min
max
count
Word Length
0
100
200
300
400
500
negative
neutral
positive
Sentiment Label

Figure 4.4

Figure 4.5 shows the word cloud of each positive, negative and neutral. Based on the positive label word cloud we can see the words ‘delivery’, ‘time’ is included so we can say that customers are satisfied with the delivery time when they order online.Next, ’will’ is a word that expresses future tense and combining with the word ‘order’ which indicating the customer will order again in the future.However, in the negative label word cloud ahad also included words ‘ delivery’ and the word ‘hour’. Some of the customers had waited for their order longer than their expected arrival time. The neutral label word cloud had been printed also as the word cloud has the similar word to negative and positive label.

Word Cloud

[41]:
```
import pandas as pd
from wordcloud import WordCloud
import matplotlib.pyplot as plt

# sentiment_df with the columns commentText and sentiment_label
positive_comments = " ".join(sentiment_df[sentiment_df.sentiment_label == 'positive'].commentText)
negative_comments = " ".join(sentiment_df[sentiment_df.sentiment_label == 'negative'].commentText)
neutral_comments = " ".join(sentiment_df[sentiment_df.sentiment_label == 'neutral'].commentText)

# create a word cloud for positive comments
wordcloud = WordCloud(width = 200, height = 200, background_color ='white', stopwords = None, min_font_size = 10).generate(positive_comments)
fig, axs = plt.subplots(1, 3, figsize=(10,5))
axs[0].imshow(wordcloud)
axs[0].axis("off")
axs[0].set_title('Positive')

# create a word cloud for negative comments
wordcloud = WordCloud(width = 200, height = 200, background_color ='white', stopwords = None, min_font_size = 10).generate(negative_comments)
axs[1].imshow(wordcloud)
axs[1].axis("off")
axs[1].set_title('Negative')

# create a word cloud for neutral comments
wordcloud = WordCloud(width = 200, height = 200, background_color ='white', stopwords = None, min_font_size = 10).generate(neutral_comments)
axs[2].imshow(wordcloud)
axs[2].axis("off")
axs[2].set_title('Neutral')

plt.tight_layout(pad = 0)
plt.show()
```

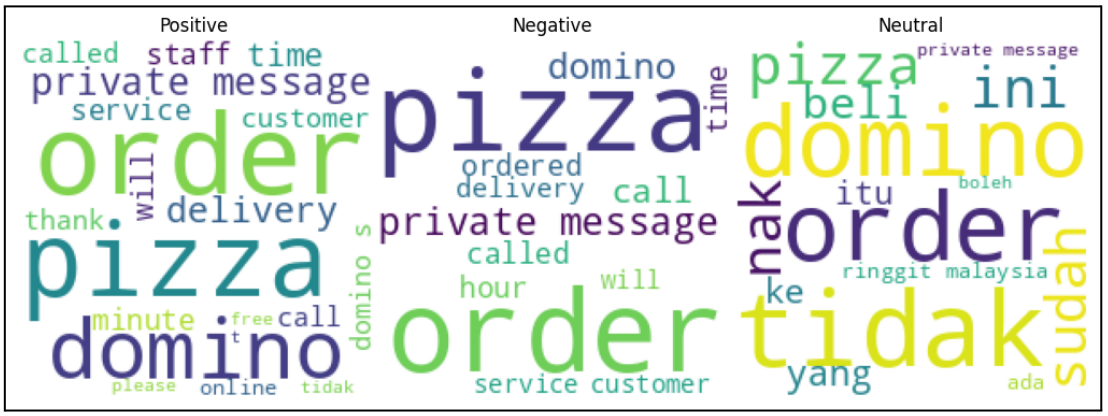


Figure 4.5

Based on the figure 4.5, the count of each sentiment label is plotted across the year. We can see that the number of comments is increasing across the year and this is the impact of COVID-19. During COVID-19 pandemic, the customers are required to order through online and the review becomes one of the tools to express their satisfaction. Also, an online customer review has become important as most customers are likely to refer to the reviews before they purchase.

Visualize total number of each sentiment label in each year

```
[42]: #how many positive/negative/neutral in each year
      # create a figure and axis
      fig, ax = plt.subplots()

      # assuming you have a dataframe called sentiment_df with the columns commentYear and sentiment_label
      grouped = sentiment_df.groupby(['commentYear', 'sentiment_label']).size().reset_index(name='count')
      pivot_table = grouped.pivot(index='commentYear', columns='sentiment_label', values='count')
      pivot_table.plot(kind='bar', ax=ax)

      # set the title and labels
      ax.set_title('Domino Sentiment Label')
      ax.set_xlabel('Comment Year')
      ax.set_ylabel('Count')
```

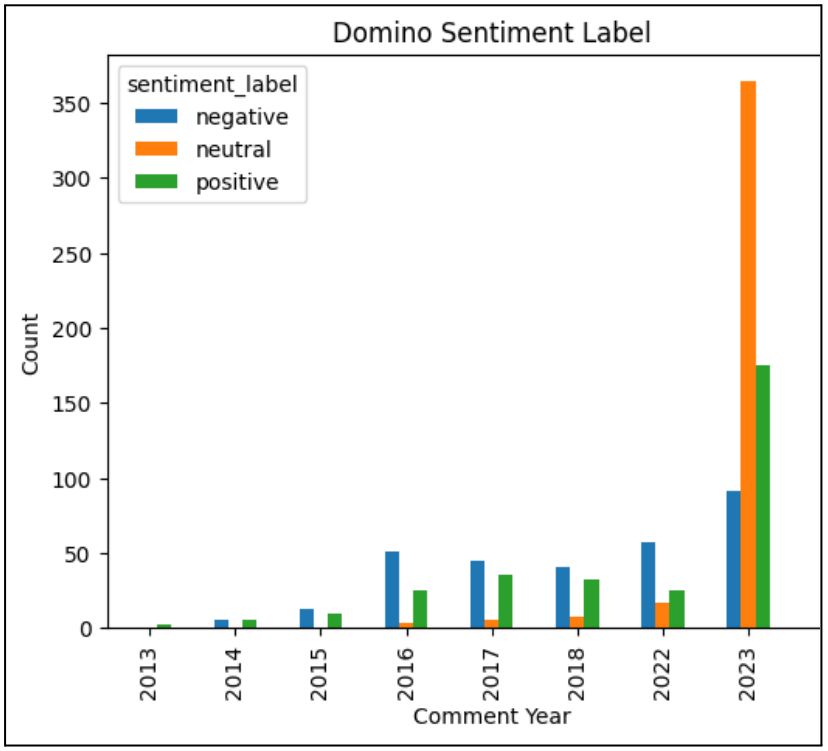


Figure 4.5

References

1. https://commentpicker.com/facebook-post-id-finder.php

2. Saleh, K. (2021). The Importance Of Online Customer Reviews [Infographic]. [online] invespcro.com. Available at: <https://www.invespcro.com/blog/the-importance-of-online-customer-reviews-infographic/> [Accessed 1 May. 2023].

3. U. (2017). FULL REPORT - 1.0 INTRODUCTION Dominos Pizza was No. 1.... [online] coursehero.com. Available at: <https://www.coursehero.com/file/23452185/FULL-REPORT/> [Accessed 30 Apr. 2023].
4. Zúñiga, K. (2022). Facebook Scraper. [online] GitHub. Available at: https://github.com/kevinzg/facebook-scraper
5. Zúñiga, K. (n.d.). facebook-scraper: Scrape Facebook public pages without an API key. [online] PyPI. Available at: https://pypi.org/project/facebook-scraper/

Appendix A Work Distribution Among Members

	Subtasks handled by each team member			
Tasks	Loke Tze Min	Sit Yie Sian	Leong Sheng Mou	Tan Jacqueline
Item 6 Extract data	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.
Item 7 Condition & transform data	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.
Item 8 NLP tool	1. 2. 3.	1. 2. 3.	1. 2. 3.	1. 2. 3.
Item 9 Text	1.	1.	1.	1.

analytics task	2. 3.	2. 3.	2. 3.	2. 3.