PDANA8411 POE PART 3

Brice Derek Agnew (ST10072411)

IIE VARSITY COLLEGE Cape Town

Contents

Introduction:

According to Marius Mogyorosim (Mogyorosi, 2025), Sentiment analysis in data analytics can be defined as the process of using Natural Language Processing and other methodologies to automatically determine the tone and emotion expressed in digital text(Mogyorosi, 2025).

In this project I will be making use of the NLTK library to perform various Sentiment analysis tasks oi order to identify the prevalent sentiment in the dataset and identify any frequently occurring concerns in the reviews.

The dataset used in this project is the publicly available "**Business Reviews from different** Industries" dataset by user **Ashlin Darius Govindasamy** on Kaggle.com(Govindasamy, 2023). I chose it because it was a relatively large dataset with around 40 000 entries, consisting of 'Hello Peter' reviews, which is the same platform and thus format that the company would use.

The types of models used:

This project made use of the **Sentiment Intensity Analyzer methodology**, an easily applicable sentiment analysis method from the 'nltk' library that doesn't require training and is easy to implement(www.nltk.com, 2025). The 'SIA' method is also very good at extracting sentiment based on punctations and emoticons, meaning that it could potentially be more accurate as it draws on more(www.nltk.com, 2025).

Logistic Regression, a classification algorithm that is well suited to the problem of sentiment analysis, it is however, better suited for labelled data and is not entirely required for the purpose of this project, I did however train one that can be used for 'Future sentiment prediction'. This additional model would prove to predict sentiment at an 86% accuracy score according to the analysis.

Plan:

This project will follow the steps of the Data Analytics process, making amendments where necessary to suit the needs of the client as described in the question paper.

The first step in this process is to define the problem. Our problem for this project is ‘How can we develop a sentiment analysis program, based on ‘Hello Peter’ reviews?’

The second step is data collection. I sourced my data as a csv file from a publicly available Kaggle dataset of hello peter reviews(Govindasamy, 2023), which aligns with our problem.

The third step is Exploratory Data Analysis, a means of visually representing and defining the types of columns in the dataset, number of missing values, number of rows, number of duplicate rows, number of missing values per column, and the datatypes of each column in the dataset.

The fourth step was Data Cleaning and Preprocessing, in which I made use of a pipeline to identify and remove any missing or duplicate values(pandas.pydata.org, 2025b), and remove any problematic columns(pandas.pydata.org, 2025a).

After data cleaning, Sentiment Analysis is undertaken using the Sentiment Intensity Analyzer package from the nltk library(www.nltk.com, 2025). This then assesses each review content to determine if the sentiment is positive, negative, or neutral. This in turn creates a label for each data entry.

Next, a Logistic regression model is trained to predict future sentiments, if necessary, for future implementation.

After the model is developed The final stage of the project is ‘Concern Identification’, in which I used the nltk ‘tokenize’(www.nltk.org, n.d.) and ‘corpus’(www.nltk.org, n.d.) packages to tokenize all the review data in the dataset, sort through them to determine the total frequency of each prominent word used in the reviews, assign each of the top 50 most frequently occurring words into 7 key categories, and determining and displaying the total frequency of each ‘concern category’ to determine which primary concerns were discussed in the reviews. According to the data analysis I undertook, the most frequently discussed concern was “Customer Service” with 127735 observed mentions in the dataset.

Report:

The preliminary step of this project is to ensure that the nltk package is installed on the local machine:

```
[1]: !pip install nltk

Requirement already satisfied: nltk in c:\users\brice\anaconda3\lib\site-packages (3.9.1)
Requirement already satisfied: click in c:\users\brice\anaconda3\lib\site-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in c:\users\brice\anaconda3\lib\site-packages (from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in c:\users\brice\anaconda3\lib\site-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in c:\users\brice\anaconda3\lib\site-packages (from nltk) (4.66.5)
Requirement already satisfied: colorama in c:\users\brice\anaconda3\lib\site-packages (from click->nltk) (0.4.6)
```

The relevant imports are added to the program to ensure access to relevant methods, functions, and models throughout the project:

```
[2]: #ST10072411 Brice Agnew
#24/06/2025
#Prog8411 POE part 3
#Sentiment Analysis


#Imports
import nltk

try:
    nltk.data.find('sentiment/vader_lexicon')
except LookupError:
    nltk.download('vader_lexicon')
# tries to find the vader_lexicon, and if not, downloads to the 'C:\Users\<user>\AppData\Roaming\nltk_data' directory

from nltk.sentiment.vader import SentimentIntensityAnalyzer as SIA
import pandas as pd
import re

from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest, chi2
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder

from sklearn.base import BaseEstimator, TransformerMixin
from wordcloud import WordCloud
import matplotlib.pyplot as plt

from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import Counter

# Download only once, ensuring that required NLTK resources are available
nltk.download('punkt')
nltk.download('stopwords')
```

After this the raw data file(Govindasamy, 2023) is imported and sved as a dataframe for use in the project:

```
•[3]: #Data Collection

#open chosen dataset csv file
#Please be sure to keep the attached csv files in the same root repository before running!
Dataset = pd.read_csv("PDANA_POE_rawdata.csv")
```

Next we can begin the Exploratory Data Analysis stage, in order to identify key aspects of the dataset:

[4]:
```python
#Exploratory Data Analysis: Part 1
Dataset.head(7)
```

[4]:

	id	user_id	created_at	authorDisplayName	author	authorAvatar	author_id	review_title	review_rating	review_content	...	industry_name
0	4228893	0cf37140-f94c-11ec-9b49-19cd672a8c8b	2022-12-16 20:51:27	Werner S	Werner S	NaN	0cf37140-f94c-11ec-9b49-19cd672a8c8b	Not telkpm	1	I pay R519 per month and can't signin on the ...	...	Telecommunication
1	4228846	364faaf0-5104-11ec-8045-7d7a9af8c558	2022-12-16 18:54:28	Nokuthula K	Nokuthula K	NaN	364faaf0-5104-11ec-8045-7d7a9af8c558	Please help me clear my name	1	I can't even buy a house because telkom failed...	...	Telecommunication
2	4228766	0c8502a0-5ee0-11ea-9899-6f17cff30ea1	2022-12-16 16:28:46	Services A	Services A	NaN	0c8502a0-5ee0-11ea-9899-6f17cff30ea1	Untrustworthy business	1	I was contacted if I wanted to upgrade a data ...	...	Telecommunication
3	4228626	35bf64d7-31fa-11e8-83f4-f23c91bb6188	2022-12-16 12:58:01	Nina H	Nina H	NaN	35bf64d7-31fa-11e8-83f4-f23c91bb6188	telkom	1	Have been in a fight with TELKOM since August ...	...	Telecommunication
4	4228610	2b490ce9-31fa-11e8-83f4-f23c91bb6188	2022-12-16 12:42:24	Arshad P	Arshad P	NaN	2b490ce9-31fa-11e8-83f4-f23c91bb6188	TELKOM Killarney Looting on Upgrade	1	-10, \nSo I have Telkom contract with Telkom f...	...	Telecommunication
5	4228533	017368a3-31fa-11e8-83f4-f23c91bb6188	2022-12-16 11:27:08	Elaine	Elaine	NaN	017368a3-31fa-11e8-83f4-f23c91bb6188	TELKOM AND NUDEBT ARE ROGUES AND FRAUDSTERS!!!	1	TELKOM ARE ROGUES. \nTHEY STEAL OUR MONEY EVER...	...	Telecommunication
6	4228398	3c07bdd1-31fa-11e8-83f4-f23c91bb6188	2022-12-16 08:48:06	Robyn S	Robyn S	NaN	3c07bdd1-31fa-11e8-83f4-f23c91bb6188	Despicable Service	1	If I could give this company a 1 I would. The ...	...	Telecommunication

7 rows × 27 columns

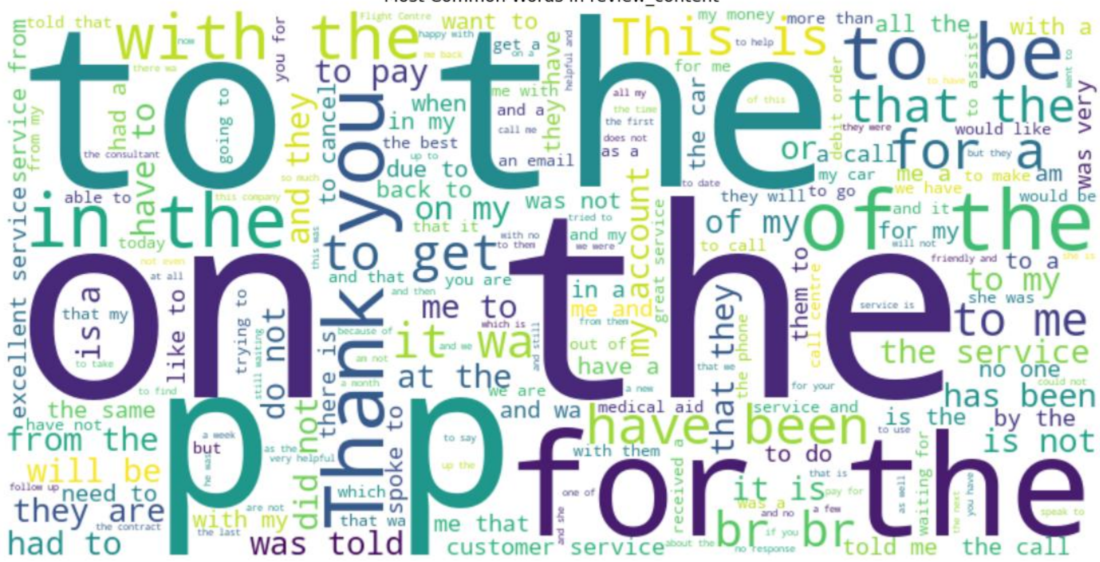
In this stage of EDA, the goal is to visually represent the properties of the dataset through visual or tabular formats to convey information. The following visual is used to proportionately represent the distribution of most frequently occurring words in the dataset:

•[5]:
```python
#Exploratory Data Analysis: Part 2

#This word cloud represents the proportional word distribution in the raw "review_content" column, which is the primary source of sentiment.
text = ' '.join(Dataset['review_content'].dropna())
wordcloud = WordCloud(width=800, height=400, background_color='white', stopwords='english').generate(text)

plt.figure(figsize=(15, 7))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title("Most Common Words in review_content")
plt.show()
```

Most Common Words in review_content



The following code identifies key properties of the dataset before cleaning, such as the number of rows, the number of duplicate values, the number of missing values in the set, and the datatypes used for each column in the raw dataset:

```
[6]: #Exploratory Data Analysis: Part 3
print("Dataset Empty value check: ")
print(Dataset.isna())
print("")

print("Number of rows:\n"
      , len(Dataset),"\n")
print("Number of Empty values:\n"
      ,Dataset.isna().any(axis=1).sum(),"\n")

#Identifying number of duplicate rows
print("Number of Duplicate rows:\n"
      ,Dataset.duplicated().sum(), "\n")


#Counting which columns have the most missing values
print("Number of missing values per column: ")
print(Dataset.isna().sum())
print("")

#investigating datatypes and column names
print("Data types per column: ")
print(Dataset.dtypes)
```

```
Dataset Empty value check:
          id  user_id  created_at  authorDisplayName  author  authorAvatar  \
0      False    False       False              False   False          True
1      False    False       False              False   False          True
2      False    False       False              False   False          True
3      False    False       False              False   False          True
4      False    False       False              False   False          True
...      ...      ...         ...                ...     ...           ...
56255  False    False       False              False   False          True
56256  False    False       False              False   False          True
56257  False    False       False              False   False          True
56258  False    False       False              False   False          True
56259  False    False       False              False   False          True

       author_id  review_title  review_rating  review_content  ...  \
0          False         False          False           False  ...
1          False         False          False           False  ...
2          False         False          False           False  ...
3          False         False          False           False  ...
4          False         False          False           False  ...
...          ...           ...            ...             ...  ...
56255      False         False          False           False  ...
56256      False         False          False           False  ...
56257      False         False          False           False  ...
56258      False         False          False           False  ...
56259      False         False          False           False  ...

       industry_name  industry_slug  status_id  nps_rating  source  \
0              False          False      False        True   False
1              False          False      False        True   False
2              False          False      False        True   False
3              False          False      False        True   False
4              False          False      False        True   False
...              ...            ...        ...         ...     ...
56255          False          False      False        True    True
56256          False          False      False        True    True
56257          False          False      False        True    True
56258          False          False      False        True    True
56259          False          False      False        True    True

       is_reported  business_reporting  author_created_date  \
0            False                True                False
1            False                True                False
2            False                True                False
3            False                True                False
4            False                True                False
...            ...                 ...                  ...
56255        False                True                False
56256        False                True                False
56257        False                True                False
56258        False                True                False
56259        False                True                False

       author_total_reviews_count  attachments
0                           False        False
1                           False        False
2                           False        False
3                           False        False
4                           False        False
...                           ...          ...
56255                       False        False
56256                       False        False
56257                       False        False
56258                       False        False
56259                       False        False

[56260 rows x 27 columns]
```

```
Number of rows:
 56260

Number of Empty values:
 56259

Number of Duplicate rows:
 0

Number of missing values per column:
id                              0
user_id                         5
created_at                      0
authorDisplayName               0
author                          0
authorAvatar                55976
author_id                       5
review_title                    6
review_rating                   0
review_content                 14
business_name                   0
business_slug                   0
permalink                       0
replied                         0
messages                        0
business_logo               11086
industry_logo                   0
industry_name                   0
industry_slug                   0
status_id                       0
nps_rating                  40745
source                      11319
is_reported                     0
business_reporting          55700
author_created_date             0
author_total_reviews_count      0
attachments                     0
dtype: int64
```

```
Data types per column:
id                              int64
user_id                        object
created_at                     object
authorDisplayName              object
author                         object
authorAvatar                   object
author_id                      object
review_title                   object
review_rating                   int64
review_content                 object
business_name                  object
business_slug                  object
permalink                      object
replied                         int64
messages                       object
business_logo                  object
industry_logo                  object
industry_name                  object
industry_slug                  object
status_id                       int64
nps_rating                    float64
source                         object
is_reported                      bool
business_reporting             object
author_created_date            object
author_total_reviews_count      int64
attachments                    object
dtype: object
```

The following code defines the key functions that will be used in the pipeline that will be used later. These functions define how the data will be cleaned and the functions selected:

```
[*]: #Data Cleamning Pipeline:
class ReviewCleaner(BaseEstimator, TransformerMixin):
    def __init__(self):
        # Define all irrelevant columns here for both Feature selection and Data Cleaning Steps
        self.cols_to_drop = [
            # Previously cleaned ones
            'business_reporting', 'authorAvatar', 'nps_rating', 'source', 'business_logo',
            # Feature selection irrelevant ones
            'id', 'user_id', 'created_at', 'messages', 'authorDisplayName', 'author',
            'author_id', 'attachments', 'industry_logo', 'industry_name', 'industry_slug',
            'is_reported', 'status_id', 'replied', 'author_created_date', 'permalink',
            'business_slug', 'business_name'
        ]

    def clean_text(self, text):
        # Remove all characters except letters, numbers, spaces, and punctuation
        return re.sub(r'[^a-zA-Z0-9\s!?.]', '', text)

    def fit(self, X, y=None):
        return self

    def transform(self, X, y=None):
        X = X.copy()

        # Drop irrelevant columns if they exist
        X.drop(columns=[col for col in self.cols_to_drop if col in X.columns], inplace=True, errors='ignore')

        # Combine and clean review_title + review_content (For sentiment analysis and Theme identification)
        if {'review_title', 'review_content'}.issubset(X.columns):
            X['review'] = (X['review_title'].fillna('') + ' ' + X['review_content'].fillna(''))
            X['review'] = X['review'].apply(self.clean_text)
            X = X.drop(columns=['review_title', 'review_content'], errors='ignore')

             X = X.drop_duplicates()
        return X

class TextSelector(BaseEstimator, TransformerMixin):
    def __init__(self, key='review'):
        self.key = key

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        return X[self.key]  # return a Series, not DataFrame
```

This code shows how the pipeline is implemented:

```
[8]: #Data Cleaning: Part 2
ml_pipeline = Pipeline([
    ('cleaner', ReviewCleaner()),  # expects a DataFrame
    ('select_text', TextSelector('review')),  # reduce to just the 'review' column
    ('vectoriser', TfidfVectorizer(stop_words='english')),
    ('select', SelectKBest(chi2, k=1000)),
    ('model', LogisticRegression(max_iter=1000))
])
```

The following is the code and output of the usage of the pipeline, the dataset is cleaned and transformed into the 'cleaned_data' dataframe, with a test output and expected output to confirm correct data transformation:

```
[9]: #Data Cleaning: Part 3
#Pipeline call
# Apply only the cleaning step
cleaned_data = ReviewCleaner().fit_transform(Dataset)

#Establishing the number of empty values remaining in the Dataset
print("Number of Empty values:\n"
      ,cleaned_data.isna().any(axis=1).sum(),"\n")

print(cleaned_data.isna().sum())
print("")

#Testing the new 'review' content
print("The review content: \n", cleaned_data['review'][0])
#Expected output: 'Not telkpm I pay R519 per month and can't signin  on the modem.  This happens when we have electricity which is about every 3 hours nc
#I would plead with potential customers not to consider telkom'
```

```
Number of Empty values:
 0

review_rating               0
author_total_reviews_count  0
review                      0
dtype: int64

The review content:
 Not telkpm I pay R519 per month and cant signin  on the modem.  This happens when we have electricity which is about every 3 hours none after 21h00
I would plead with potential customers not to consider telkom
```

The next section of code demonstrates how the data has been transformed by displaying the number of rows, empty values, duplicate values, and columns that are present in the cleaned dataset, in comparison to the raw dataset:

```
[10]: #Data Cleaning: Part 4
#visually representing changes
print("Number of rows:\n"
      , len(cleaned_data),"\n")
print("Number of Empty values:\n"
      ,cleaned_data.isna().any(axis=1).sum(),"\n")

#Identifying number of duplicate rows
print("Number of Duplicate rows:\n"
      ,cleaned_data.duplicated().sum(), "\n")

#Counting which columns have the most missing values
print(cleaned_data.isna().sum())
print("")
```

```
Number of rows:
 55926

Number of Empty values:
 0

Number of Duplicate rows:
 0

review_rating               0
author_total_reviews_count  0
review                      0
dtype: int64
```

This following section of code shows how the Sentiment Intensity Analyzer module is used in the program to assign levels of sentiment to each row, as well as the columns that are created in this process:

[11]:

```
#Sentiment Analysis:
#using the SentimentIntensityAnalyzer package from the NLTK library, specifically the vader
#SIA doesnt require training, which does make the process simpler. Additionally it is reported to be good at interpretting punvctuation in sentiment.

#calculate sentiment
analyzer = SIA()
cleaned_data['compound'] = [analyzer.polarity_scores(x)['compound'] for x in cleaned_data['review']]
cleaned_data['negative'] = [analyzer.polarity_scores(x)['neg'] for x in cleaned_data['review']]
cleaned_data['neutral'] = [analyzer.polarity_scores(x)['neu'] for x in cleaned_data['review']]
cleaned_data['positive'] = [analyzer.polarity_scores(x)['pos'] for x in cleaned_data['review']]

#create columns
cleaned_data['sentiment']='neutral' #Automatically setting 'overall' sentiment to being neutral before individually assigning positive or negative
cleaned_data.loc[cleaned_data.compound>0.05,'sentiment']='positive'
cleaned_data.loc[cleaned_data.compound<-0.05,'sentiment']='negative'
cleaned_data.head()

#on-screen summary
print("Neutral Total Sentiment Count: ",
      cleaned_data['sentiment'].value_counts()['neutral'])
print("Positive Total Sentiment Count: ",
      cleaned_data['sentiment'].value_counts()['positive'])
print("Negative Total Sentiment Count: ",
      cleaned_data['sentiment'].value_counts()['negative'])

cleaned_data.head(5)
```

```
Neutral Total Sentiment Count:  1884
Positive Total Sentiment Count:  34192
Negative Total Sentiment Count:  19850
```

[11]:

	review_rating	author_total_reviews_count	review	compound	negative	neutral	positive	sentiment
0	1	3	Not telkpm I pay R519 per month and cant signi...	0.0762	0.000	0.963	0.037	positive
1	1	3	Please help me clear my name I cant even buy a...	0.8608	0.055	0.768	0.177	positive
2	1	50	Untrustworthy business I was contacted if I wa...	-0.5994	0.079	0.890	0.031	negative
3	1	1	telkom Have been in a fight with TELKOM since ...	-0.9672	0.216	0.706	0.078	negative
4	1	5	TELKOM Killarney Looting on Upgrade 10 \nSo I ...	-0.9656	0.101	0.861	0.038	negative

The following code shows how a Logistic regression model is trained in order to predict the sentiment labels that were previously defined in the last sections, the model was then evaluated based on the metrics of Precision, Recall, f1-scores, and support.

•[12]:
```python
#Model training:
required_columns = {'review', 'sentiment'}

# Check if required columns exist
if not required_columns.issubset(cleaned_data.columns):
    missing = required_columns - set(cleaned_data.columns)
    raise ValueError(f"Columns missing from dataset: {missing}")

# Proceed only if columns exist
cleaned_data = cleaned_data.dropna(subset=['review', 'sentiment'])
#cleaned_data= cleaned_data.drop(columns=['review_title', 'review_content'], errors='ignore')

X = cleaned_data  # needed by ReviewCleaner
y = cleaned_data['sentiment']

# Convert labels to numeric if needed
le = LabelEncoder()
y_encoded = le.fit_transform(y)

print("Length of X:", len(X))
print("Length of y:", len(y_encoded))

# Split data
#X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=69)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=69)

# Fit model
ml_pipeline.fit(X_train, y_train)

# Predict
preds = ml_pipeline.predict(X_test)

# Evaluate
print(classification_report(y_test, preds, target_names=le.classes_))

cleaned_data.head()
```

```
Length of X: 55926
Length of y: 55926
              precision    recall  f1-score   support

    negative       0.80      0.90      0.85      3934
     neutral       0.82      0.12      0.21       354
    positive       0.91      0.89      0.90      6898

    accuracy                           0.87     11186
   macro avg       0.85      0.63      0.65     11186
weighted avg       0.87      0.87      0.86     11186
```

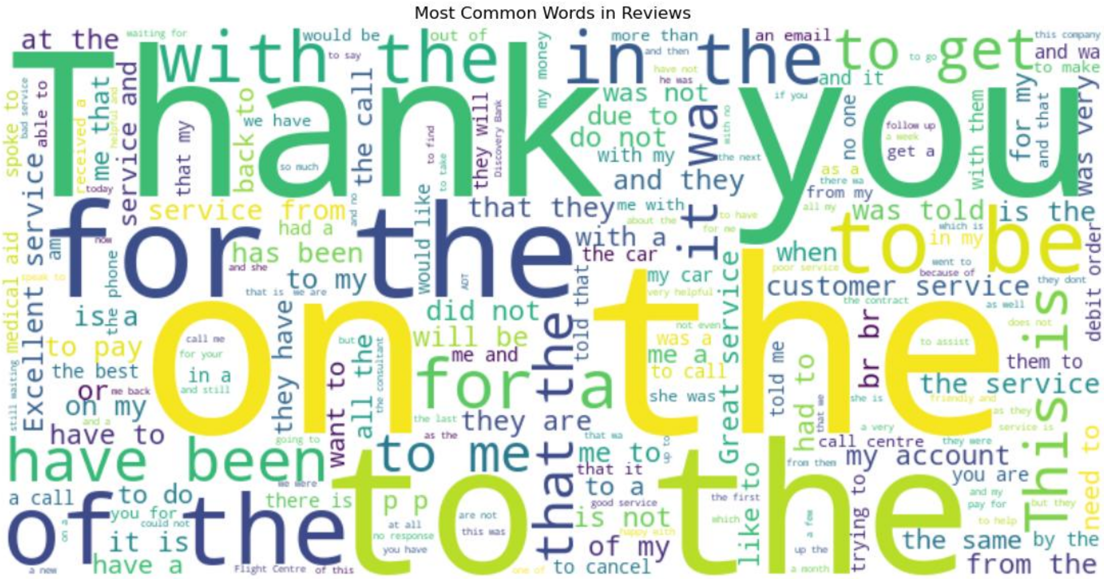
[12]:

	review_rating	author_total_reviews_count	review	compound	negative	neutral	positive	sentiment
0	1	3	Not telkpm I pay R519 per month and cant signi...	0.0762	0.000	0.963	0.037	positive
1	1	3	Please help me clear my name I cant even buy a...	0.8608	0.055	0.768	0.177	positive
2	1	50	Untrustworthy business I was contacted if I wa...	-0.5994	0.079	0.890	0.031	negative
3	1	1	telkom Have been in a fight with TELKOM since ...	-0.9672	0.216	0.706	0.078	negative
4	1	5	TELKOM Killarney Looting on Upgrade 10 \nSo I ...	-0.9656	0.101	0.861	0.038	negative

The following code section is used to develop a visual representation of word distribution in the cleaned dataset:

```
[13]: #Visualization of word frequency in reviews
      text = ' '.join(cleaned_data['review'].dropna())
      wordcloud = WordCloud(width=800, height=400, background_color='white', stopwords='english').generate(text)

      plt.figure(figsize=(15, 7))
      plt.imshow(wordcloud, interpolation='bilinear')
      plt.axis('off')
      plt.title("Most Common Words in Reviews")
      plt.show()
```



The following code is used to tokenise every word used in the reviews in the dataset, and rank them based on most frequently used, ranking the top 50 most used words:

[14]:

```python
#Concern Identification: Part 1
#Most frequently occuring words/concerns

# Prepare stopwords as a set for faster lookup
stop_words = set(stopwords.words('english'))

# Initialize counter
token_counts = Counter()

# Iterate over each review
for review in cleaned_data['review'].dropna().astype(str):
    tokens = word_tokenize(review.lower())
    tokens = [t for t in tokens if t.isalpha() and t not in stop_words]
    token_counts.update(tokens)
#the first version I tried with this involved compiling every review into a string to search through, but that had insane levels of overhead
#so this secondary version was developed to instead use a counter object and for loop to more efficiently search through the tokens found.

#Uses the token_count counter data to create a dataframe for display.
freq_df = pd.DataFrame(token_counts.items(), columns=['word', 'frequency'])
freq_df['frequency'] = freq_df['frequency'].astype(int)
freq_df = freq_df.sort_values(by='frequency', ascending=False)

print("Most frequently occuring words:")
freq_df.head(50)
```

```
Most frequently occuring words:
```

[14]:

	word	frequency
127	service	43802
153	call	16070
70	get	15030
12	would	12597
219	time	12015
61	still	11223
499	received	11032
162	told	10778
1975	thank	10677
392	back	10490
81	account	10152
420	one	10017
17	telkom	9941
1180	great	9428
22	even	9258
209	customer	9115
1693	car	8903
221	called	8822
384	good	8648
5345	br	8233
1	pay	7933
195	money	7897
5747	excellent	7833
255	dont	7832

This final section of code takes the data insights extracted from the previous section and manually assigns each of the most frequently used top fifty words into different categories of concerns, with it then counted the occurrences of to determine the ranking of discussing themes in the reviews overall:

•[15]:
```python
#Concern Identification: Part 2
#Manually seperating concerns into themes, and identifying the most commonly occuring themes

themes = {
    'Billing' : ['pay', 'money', 'account', 'paid', 'bank', 'number', 'debit','bill'],
    'Customer Service':['call', 'called', 'told', 'said', 'email', 'customer', 'help', 'please', 'service', 'sent', 'br'],
    'Product' :['telkom', 'company', 'network', 'phone', 'work', 'internet'],
    'Timeliness':['time', 'days', 'day', 'month', 'still', 'back', 'never'],
    'Complaints':['dont', 'need', 'want', 'im', 'never', 'get', 'like', 'made', 'one'],
    'Positive Sentiment':['thank', 'great', 'good', 'excellent'],
    'Ambiguous':['br','p']
}
theme_counts = {}

#iterating through the previous frequency distribution to assign each keyword to a theme
for theme, keywords in themes.items():
    total = sum(freq_df[freq_df['word'].isin(keywords)]['frequency'])
    theme_counts[theme] = total

theme_df = pd.DataFrame(list(theme_counts.items()), columns=['Concern', 'Mentions'])

#Presenting the most commonly occuring
print("[Most commonly occuring concerns:]")

theme_df.head(20).sort_values(by='Mentions', ascending=False)
```

[Most commonly occuring concerns:]

[15]:

	Concern	Mentions
1	Customer Service	127479
4	Complaints	71285
3	Timeliness	60504
0	Billing	46656
5	Positive Sentiment	36586
2	Product	31441
6	Ambiguous	13674

The final section in the project contains references to that were used in informing the creation of this project:

[16]:
```python
#References:

#Govindasamy, A.D., 2023. Business Reviews from different Industries. [online] kaggle.com. Available at: <https://www.kaggle.com/datasets/ashlingovindasc
#Mogyorosi, M., 2025. Sentiment Analysis: First Steps With Python’s NLTK Library. [online] realpython.com. Available at: <https://realpython.com/python-r
#pandas.pydata.org, 2025a. pandas.DataFrame.drop. [online] pandas.pydata.org. Available at: <https://pandas.pydata.org/pandas-docs/stable/reference/api/p
#pandas.pydata.org, 2025b. pandas.DataFrame.dropna. [online] pandas.pydata.org. Available at: <https://pandas.pydata.org/pandas-docs/stable/reference/api
#www.nltk.com, 2025. nltk.sentiment.SentimentIntensityAnalyzer. [online] www.nltk.com. Available at: <https://www.nltk.org/api/nltk.sentiment.SentimentIr
#www.nltk.org, n.d. nltk.corpus package. www.nltk.org. [online] Available at: <https://www.nltk.org/api/nltk.corpus.html> [Accessed 27 June 2025a].
#www.nltk.org, n.d. nltk.tokenize package. www.nltk.org. [online] Available at: <https://www.nltk.org/api/nltk.tokenize.html> [Accessed 27 June 2025b].
```

References

Govindasamy, A.D., 2023. *Business Reviews from different Industries*. [online] kaggle.com. Available at: <https://www.kaggle.com/datasets/ashlingovindasamy/business-reviews-from-different-industries?select=trainingData.csv> [Accessed 24 June 2025].

Mogyorosi, M., 2025. *Sentiment Analysis: First Steps With Python's NLTK Library*. [online] realpython.com. Available at: <https://realpython.com/python-nltk-sentiment-analysis/> [Accessed 24 June 2025].

pandas.pydata.org, 2025a. *pandas.DataFrame.drop*. [online] pandas.pydata.org. Available at: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.drop.html> [Accessed 24 June 2025].

pandas.pydata.org, 2025b. *pandas.DataFrame.dropna*. [online] pandas.pydata.org. Available at: <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dropna.html> [Accessed 24 June 2025].

www.nltk.com, 2025. *nltk.sentiment.SentimentIntensityAnalyzer*. [online] www.nltk.com. Available at: <https://www.nltk.org/api/nltk.sentiment.SentimentIntensityAnalyzer.html?highlight=sentimentintensity> [Accessed 27 June 2025].

www.nltk.org, n.d. nltk.corpus package. *www.nltk.org*. [online] Available at: <https://www.nltk.org/api/nltk.corpus.html> [Accessed 27 June 2025a].

www.nltk.org, n.d. nltk.tokenize package. *www.nltk.org*. [online] Available at: <https://www.nltk.org/api/nltk.tokenize.html> [Accessed 27 June 2025b].