Cambodia University of Technology and Science (CamTech)

Sentiment Analysis on Customers' Comments

By Ven Seyhah, PhD

2025-09-19

How can AI be integrated in the research, data analysis, data science, and reporting?

Just ask AI to write the whole report for us?
-> Hallucination

Introduction

This session present a hand-on workflow to analyze customer comments using an AI model (**gpt-oss:20b via Ollama**).

The pipeline performs the following steps:

1. Load raw comments data.
2. Run sentiment classification (**Positive, Negative, Neutral**).
3. Visualize the distribution of sentiments.
4. Generate an AI-powered summary of the results.

Github repository:https://github.com/seyhah/mock_lesson/tree/master#

Source code: https://github.com/seyhah/mock_lesson.git

Step 1 — Import Libraries

We begin by importing the required Python libraries:

- `pandas` for handling and manipulating structured data (tables).
- `matplotlib` for plotting and visualizing the results.
- `ollama` to connect and interact with the local AI model(`gpt-oss:20b`).
- `great_table` to present table in styled format.
- `textwrap` to the summary of the results.
- `re` to clean text.
- `Markdown, display` to allows rendering HTML/Markdown directly in a notebook cell.

```
import pandas as pd
import matplotlib.pyplot as plt
from ollama import chat
import textwrap
import re
import great_tables as gt
from IPython.display import Markdown, display
```

Step 2 — Load Data

We load customer comments stored in **comments.csv**.
Each row contains one comment.

```
df = pd.read_csv("comments.csv")
df.columns = [col.capitalize() for col in df.columns]
# Show the first 5 rows as a styled table
gt.GT(df.head()).tab_options(
    table_font_size="30px",      # increase font size
    data_row_padding="15px",      # add more space between rows
    column_labels_padding="10px" # bigger headers
)
```

Comment
The product quality is amazing and I'm very satisfied!
Customer service was slow and unhelpful.
Delivery was on time, nothing special.
The app keeps crashing and it's frustrating.
Great prices and very fast shipping.

Step 3 — Sentiment Analysis with Ollama

Here we run AI-powered sentiment analysis on each comment.

Explanation of the code:

1. We loop through every comment in the dataset.
2. For each comment, we create a prompt asking the AI model to classify it as *Positive, Negative, or Neutral.*
3. The AI's response is captured using `chat()`.
4. The classified sentiment is stored along with the original comment in a results list.
5. We convert the results list into a DataFrame and save it to `sentiment_results.csv`.

Step 3: Code

```
results = []
for comment in df['Comment']:
    prompt = f"You are an expert in Sentiment Analysis on customer’s comments and feedback. 
    resp = chat(model="gpt-oss:20b", messages=[{"role": "user", "content": prompt}])
    sentiment = resp['message']['content'].strip()

    # --- Cleaning step ---
    # Remove formatting like **text**, extra spaces, lowercase everything
    sentiment = re.sub(r"[^a-zA-Z]", "", sentiment).lower()
    # Normalize variations
    if "pos" in sentiment:
        sentiment = "Positive"
    elif "neg" in sentiment:
        sentiment = "Negative"
    elif "neu" in sentiment:
        sentiment = "Neutral"
    else:
        sentiment = "Unknown"

    results.append({"comment": comment, "sentiment": sentiment})

sentiment_df = pd.DataFrame(results)
sentiment_df.to_csv("sentiment_results.csv", index=False)
```

Step 3: Results

```
sentiment_df.columns = [col.capitalize() for col in sentiment_df.columns]
gt.GT(sentiment_df.head()).tab_options(
    table_font_size="30px",      # increase font size
    data_row_padding="15px",      # add more space between rows
    column_labels_padding="10px" # bigger headers
)
```

Comment	Sentiment
The product quality is amazing and I'm very satisfied!	Positive
Customer service was slow and unhelpful.	Negative
Delivery was on time, nothing special.	Neutral
The app keeps crashing and it's frustrating.	Negative
Great prices and very fast shipping.	Positive

Step 4 — Reload Processed Sentiment Data

We reload the processed results from `sentiment_results.csv`.
This ensures we always work with the cleaned dataset (*comment* + *sentiment*).

```
df = pd.read_csv("sentiment_results.csv")
df.columns = [col.capitalize() for col in df.columns]
gt.GT(df.head()).tab_options(
    table_font_size="30px",      # increase font size
    data_row_padding="15px",      # add more space between rows
    column_labels_padding="10px" # bigger headers
)
```

Comment	Sentiment
The product quality is amazing and I'm very satisfied!	Positive
Customer service was slow and unhelpful.	Negative
Delivery was on time, nothing special.	Neutral
The app keeps crashing and it's frustrating.	Negative
Great prices and very fast shipping.	Positive

Step 5 — Visualize Sentiment Distribution

Now we visualize the sentiment breakdown using a bar chart.

Explanation of the code:

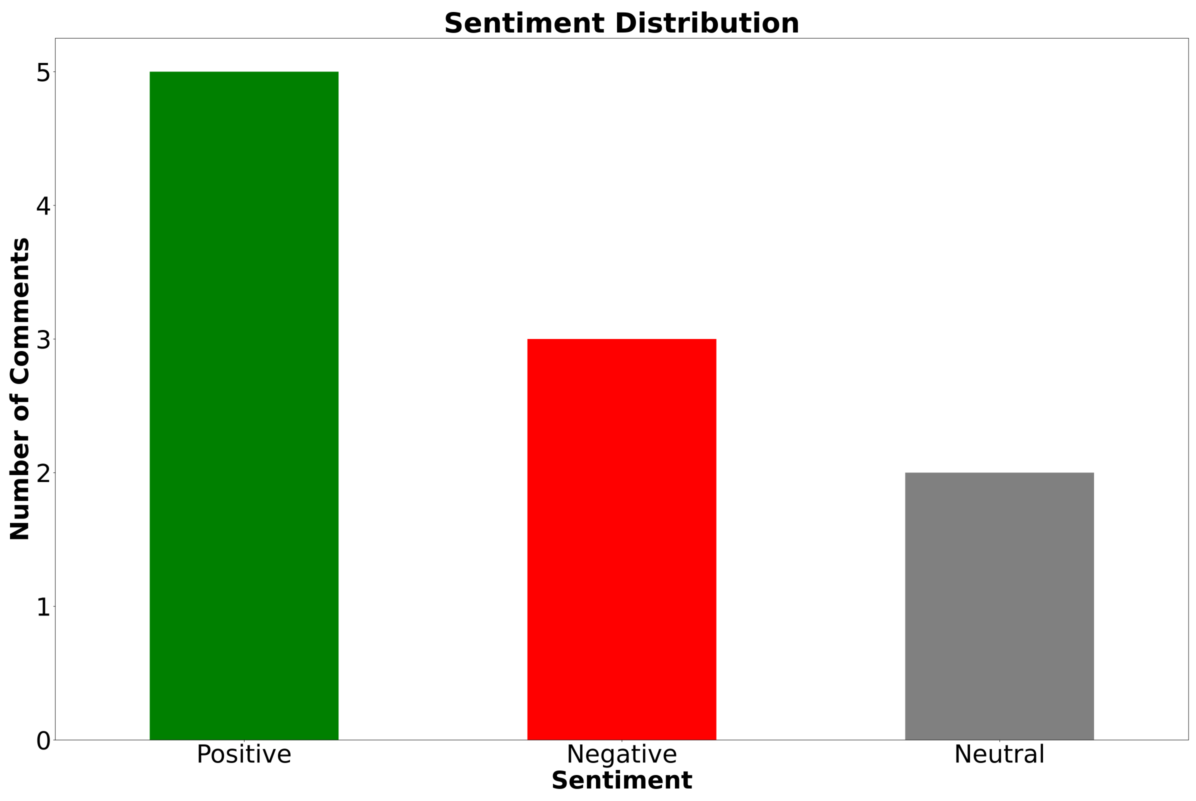
1. `value_counts()` counts how many comments fall into each sentiment category.
2. We create a bar chart with:
   - **Green → Positive**
   - **Red → Negative**
   - **Gray → Neutral**
3. Labels and titles are added to make the chart more readable.

Step 5: Code, Function for plotting

```python
import matplotlib.pyplot as plt

def plot_sentiment_distribution(df, column='Sentiment'):
    """
    Plot the sentiment distribution from a dataframe column.

    Parameters:
        df (pd.DataFrame): Input dataframe containing sentiment column.
        column (str): Name of the column with sentiment labels.
    """
    sentiment_counts = df[column].value_counts()

    plt.figure(figsize=(30, 20))
    sentiment_counts.plot(
        kind='bar',
        color=['green', 'red', 'gray'][:len(sentiment_counts)]
    )
    plt.title("Sentiment Distribution", fontsize=50, weight='bold')
    plt.xlabel("Sentiment", fontsize=44, weight='bold')
    plt.ylabel("Number of Comments", fontsize=45, weight='bold')
    plt.xticks(rotation=0, fontsize=45)
    plt.yticks(fontsize=45)
    plt.tight_layout()
    plt.show()
```

Step 5: Bar Chart

```python
plot_sentiment_distribution(df)
```



Step 6 — AI Summary of Sentiment Results

We ask the AI model to generate a short natural-language summary of the sentiment analysis results.

Explanation of the code:

1. We pass the sentiment counts as context to the model.

2. The model generates a summary describing the overall sentiment distribution.

3. The result is printed for quick interpretation.

Step 6: Code

```
sentiment_counts = df['Sentiment'].value_counts()

prompt = f"""
You are a research assistant in summarizing the results of a sentiment analysis task.
Here are the sentiment counts from customer comments:
{sentiment_counts.to_dict()}

Write a short summary of these results.
- Use the exact numbers given (do not approximate).
- Mention Positive, Negative, and Neutral counts explicitly.
- Keep the explanation concise.
- Add an interpretation of each number
- Add a conclosion sentence at the end.
"""

response = chat(model="gpt-oss:20b", messages=[{"role": "user", "content": prompt}])
summary = response['message']['content']

'''
# --- Clean the summary text ---
clean_summary = summary
clean_summary = re.sub(r"\*\*(.*?)\*\*", r"\1", clean_summary)  # remove bold markdown
clean_summary = re.sub(r"[ ~]", "", clean_summary)              # remove symbols
clean_summary = re.sub(r"\s+", " ", clean_summary).strip()      # normalize spaces
'''
# Remove common dash variations and extra spaces
clean_summary = re.sub(r"[---]", " ", summary)   # replace -, -, - with space
clean_summary = re.sub(r"\s+", " ", clean_summary).strip()  # normalize spaces

# Wrap for display
wrapped_summary = textwrap.fill(clean_summary, width=80)
```

```
<>:22: SyntaxWarning: invalid escape sequence '\*'
<>:22: SyntaxWarning: invalid escape sequence '\*'
C:\Users\User\AppData\Local\Temp\ipykernel_24060\2418319423.py:22: SyntaxWarning: invalid esc
  clean_summary = re.sub(r"\*\*(.*?)\*\*", r"\1", clean_summary)  # remove bold markdown
```

Step 6: Summary of Results

```
display(Markdown(wrapped_summary))
```

The sentiment analysis of the customer comments yielded the following counts: **Positive: 5**, **Negative: 3**, **Neutral: 2**. **Positive (5)**: A majority of the feedback is favorable, suggesting that most customers are satisfied with the product or service. **Negative (3)**: A noticeable minority expressed dissatisfaction, indicating areas that may need improvement. **Neutral (2)**: A small group had neither positive nor negative feelings, reflecting ambivalence or lack of strong opinion. Overall, the data shows a slightly positive customer sentiment.

Conclusion

This analysis demonstrates how **AI + Python + Quarto** can be combined to:

- Automate text classification (*sentiment detection*).
- Generate visual summaries using charts.
- Provide natural-language insights with AI.

Such a workflow can be applied to any text-based dataset (e.g., surveys, feedback, reviews) to quickly extract actionable business intelligence.

This workflow can also be automated to produce reports periodically with a single click.