Cambodia University of Technology and Science (CamTech)

Customer Comments Sentiment Analysis

By: Ven Seyhah, PhD

Date: 19 Setember 2025

This notebook analyzes 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.

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).

```
import pandas as pd
import matplotlib.pyplot as plt
from ollama import chat
```

Step 2 — Load Data

We load customer comments stored in `comments.csv`. * Each row contains one comment. * df.head() displays the first few rows so we can confirm the data is loaded correctly.

```
df = pd.read_csv("comments.csv")
df.head()
```

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 (sentiment_df) and save it to sentiment_results.csv for later use.

```
results = []

for comment in df['comment']:
    prompt = f"Classify this customer comment as Positive, Negative, or Neutral:\n\n{comment
    resp = chat(model="gpt-oss:20b", messages=[{"role": "user", "content": prompt}])
    sentiment = resp['message']['content'].strip()
    results.append({"comment": comment, "sentiment": sentiment})

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

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.head()
```

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.

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

plt.figure(figsize=(6,4))
sentiment_counts.plot(kind='bar', color=['green','red','gray'])
plt.title("Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Number of Comments")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```

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.

```
prompt = f"""
Here are the sentiment counts from customer comments:
{sentiment_counts.to_dict()}
Write a short summary describing the results.
"""

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

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 be used to automate the production of reports periodically with a click.