

AI ASSISTED CODING

ASSIGNMENT-4.5

M.AKHIL REDDY

2303A52315

BATCH-45

Objective: To explore and compare Zero-shot, One-shot, and Few-shot prompting techniques for classifying emails into predefined categories using a large language model (LLM).

1. Suppose that you work for a company that receives hundreds of customer emails daily. Management wants to automatically classify emails into categories like "Billing", "Technical Support", "Feedback", and "Others" before assigning them to appropriate departments.

Instead of training a new model, your task is to use prompt engineering techniques with an existing LLM to handle the classification.

Tasks to be completed are as below

a. Prepare Sample Data:

- Create or collect 10 short email samples, each belonging to one of the 4 categories.

b. Zero-shot Prompting:

- Design a prompt that asks the LLM to classify a single email without providing any examples.

• Example prompt:

"Classify the following email into one of the following categories:

Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month.'

c. One-shot Prompting:

- Add one labeled example before asking the model to classify a new email.

d. Few-shot Prompting:

- Use 3–5 labeled examples in your prompt before asking the model to classify a new email.

e. Evaluation:

- Run all three techniques on the same set of 5 test emails.
- Compare and document the accuracy and clarity of responses.

PROMPTS:

Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Provide only the category name as your response.

Email: [EMAIL_TEXT]

Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Provide only the category name as your response.

Example:

Email: "I was charged twice for my subscription this month. Please refund the extra amount."

Category: Billing

Email: [EMAIL_TEXT]

Classify the following email into one of the following categories: Billing, Technical Support, Feedback, Others. Provide only the category name as your response.

Example:

Email: "I was charged twice for my subscription this month. Please refund the extra amount."

Category: Billing

Example:

Email: "My app keeps crashing when I try to login. Can you help fix this?"

Category: Technical Support

Example:

Email: "I love the new interface update! It's much easier to navigate."

Category: Feedback

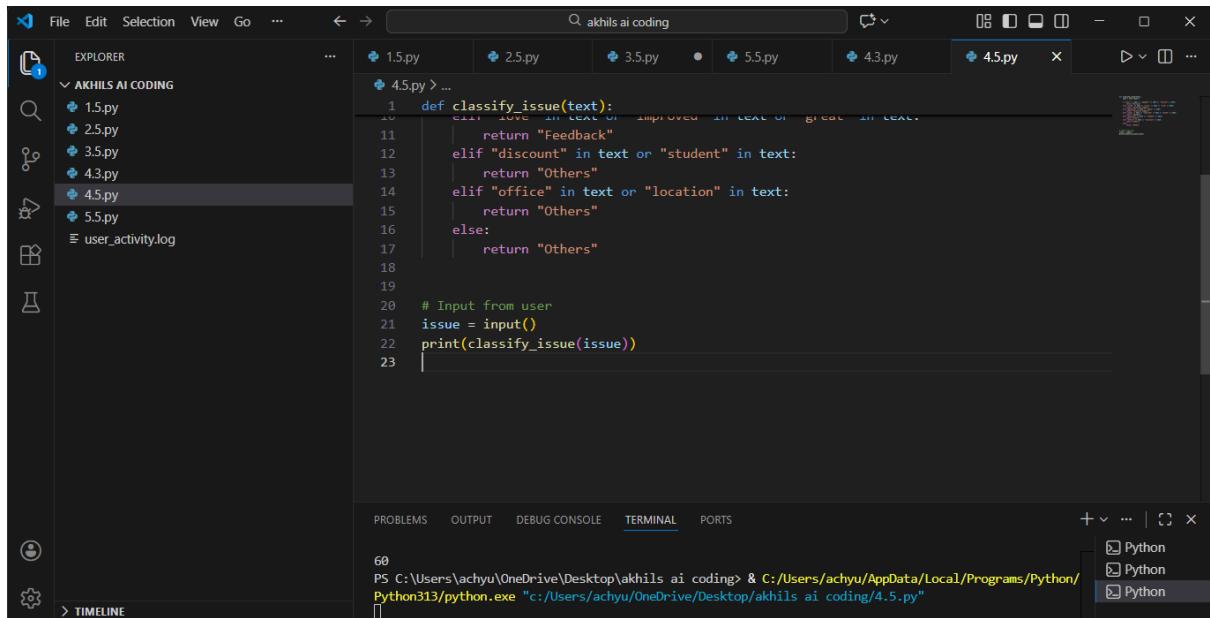
Example:

Email: "What are your office hours? I'd like to visit for a meeting."

Category: Others

Email: [EMAIL_TEXT]

EXPECTED OUTPUT:



A screenshot of the Visual Studio Code (VS Code) interface. The Explorer sidebar on the left shows a folder named 'AKHILS AI CODING' containing several Python files: 1.5.py, 2.5.py, 3.5.py, 4.3.py, 4.5.py, and 5.5.py. A file named 'user_activity.log' is also listed. The main code editor window displays a Python script named '4.5.py'. The script defines a function 'classify_issue(text)' that checks for specific keywords ('love', 'improved', 'great', 'discount', 'student', 'office', 'location') and returns a category ('Feedback', 'Others'). It also includes a comment about input from the user and a print statement. Below the code editor, the 'TERMINAL' tab is active, showing a command-line session in PowerShell (PS) with the path 'C:\Users\achyu\OneDrive\Desktop\akhils ai coding & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.5.py"'. The terminal output shows the number '60'.

```
def classify_issue(text):
    if "love" in text or "improved" in text or "great" in text:
        return "Feedback"
    elif "discount" in text or "student" in text:
        return "Others"
    elif "office" in text or "location" in text:
        return "Others"
    else:
        return "Others"

# Input from user
issue = input()
print(classify_issue(issue))
```

2. Travel Query

Classification Scenario:

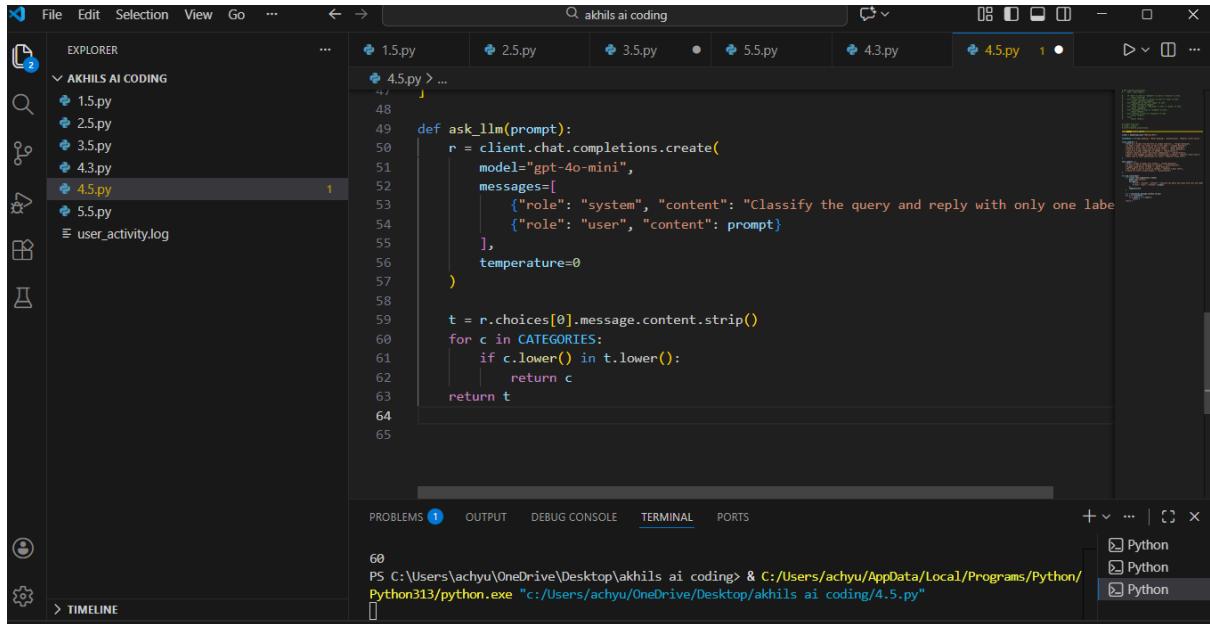
A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info.

Tasks:

- a. Prepare labeled travel queries.
- b. Apply Zero-shot prompting.
- c. Apply One-shot prompting.
- d. Apply Few-shot prompting.

e. Compare response consistency.

Expected output:



```
def ask_llm(prompt):
    r = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": "Classify the query and reply with only one label."},
            {"role": "user", "content": prompt}
        ],
        temperature=0
    )

    t = r.choices[0].message.content.strip()
    for c in CATEGORIES:
        if c.lower() in t.lower():
            return c
    return t
```

3. Programming Question Type Identification

Scenario:

A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question.

Tasks:

- a. Prepare coding-related user queries.
- b. Perform Zero-shot classification.
- c. Perform One-shot classification.
- d. Perform Few-shot classification.
- e. Analyze improvements in technical accuracy.

EXPECTED OUTPUT:

A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows an 'EXPLORER' view with a folder named 'AKHILS AI CODING' containing files 1.5.py, 2.5.py, 3.5.py, 5.5.py, and 4.5.py. The file 4.5.py is currently selected and open in the main editor area. The code in 4.5.py is a Python script for classifying text using an AI model. The right side of the interface includes a 'PROBLEMS' panel with one error, an 'OUTPUT' panel showing a command-line session, and a 'TERMINAL' panel where the script is being run. A status bar at the bottom provides details about the file and environment.

```
89 ]
90
91 def ask_llm(prompt):
92     r = client.chat.completions.create(
93         model="gpt-4o-mini",
94         messages=[
95             {"role": "system", "content": "Classify the coding query and return only one line of text."},
96             {"role": "user", "content": prompt}
97         ],
98         temperature=0
99     )
100
101 t = r.choices[0].message.content.strip()
102 for c in CATEGORIES:
103     if c.lower() in t.lower():
104         return c
105
106
107
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\achyu\OneDrive\Desktop\akhils ai coding & C:/Users/achyu/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/achyu/OneDrive/Desktop/akhils ai coding/4.5.py"
Ln 106, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.13.7 ⓘ Go Live

4. Social Media Post Categorization

Scenario:

A social media analytics tool must classify posts into Promotion, Complaint, Appreciation, or Inquiry.

Tasks:

1. Prepare sample social media posts.
2. Use Zero-shot prompting.
3. Use One-shot prompting.
4. Use Few-shot prompting.
5. Analyze informal language handling.

EXPECTED OUTPUT:

A screenshot of the Visual Studio Code (VS Code) interface. The window title is "akhils ai coding". The left sidebar shows a file tree with a folder named "AKHILS AI CODING" containing files 1.5.py, 2.5.py, 3.5.py, 4.3.py, 4.5.py (which is currently selected), and 5.5.py. Below the file tree is a "user_activity.log" file. The main editor area displays a Python script with code for interacting with a large language model (LLM). The terminal at the bottom shows the command "python 4.5.py" being run, and the output indicates the script is classifying the input query. The status bar at the bottom right shows the date and time as "04-02-2026".

```
def ask_llm(prompt):
    r = client.chat.completions.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": "Classify the coding query and return only one line of code."},
            {"role": "user", "content": prompt}
        ],
        temperature=0
    )

    t = r.choices[0].message.content.strip()
    for c in CATEGORIES:
        if c.lower() in t.lower():
            return c
    return t
```