

Lab Assignment – 4.5

Hall Ticket No.: 2303A510D9

Bt– 29

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.

Evaluation:

- Run all three techniques on the same set of 5 test emails.

- Compare and document the accuracy and clarity of responses.

```

File Edit Selection View ... ← → Search
RUN AND DEBUG RUN Welcome aac45.py
C:\> Users > boora > Downloads > aac45.py > travel_few_shot
1 # zero-shot, One-shot, Few-shot
2
3 def llm_simulator(prompt):
4     """
5         This function simulates an LLM response.
6         In real applications, this is where API calls go.
7     """
8
9     print("\n--- PROMPT SENT TO MODEL ---")
10    print(prompt)
11    print("-----")
12
13    # Dummy output for assignment
14    return "Predicted Category"
15
16
17 # 1. EMAIL CLASSIFICATION
18
19 def email_zero_shot(email):
20     prompt = f"""
21     Classify the email into:
22     Billing, Technical Support, Feedback, Others
23
24     Email: {email}
25     Category:
26     """
27     return llm_simulator(prompt)
28
29
30 def email_one_shot(email):
31     prompt = f"""
32     Example:
33     Email: My payment was deducted twice.
34     Category: Billing
35
36     Now classify:
37     Email: {email}
38     Category:
39     """
40     return llm_simulator(prompt)
41
42
43 def email_few_shot(email):
44     prompt = f"""
45     Example 1:
46     Email: I was charged extra.
47     Category: Billing
48
49     Example 2:
50     Email: App crashes on login.
51     Category: Technical Support
52
53     Example 3:
54     Email: Great service!
55     Category: Feedback
56
57     Now classify:
58     Email: {email}
59     Category:
60     """
61     return llm_simulator(prompt)
62
63
64
65 # 2. TRAVEL QUERY CLASSIFICATION
66
67 def travel_few_shot(query):
68

```

The screenshot shows a code editor window with a Python file named 'aac45.py'. The code defines three functions: 'llm_simulator', 'email_zero_shot', and 'email_few_shot'. The 'email_zero_shot' function takes an email as input and returns a category. It includes a multi-line string 'prompt' with examples for 'Billing', 'Technical Support', 'Feedback', and 'Others'. The 'email_one_shot' function uses a single example ('My payment was deducted twice.') and asks for classification. The 'email_few_shot' function provides three examples ('I was charged extra.', 'App crashes on login.', 'Great service!') and asks for classification. The 'travel_few_shot' function is partially defined at the bottom. The interface includes a 'RUN AND DEBUG' sidebar, a 'Breakpoints' section with checkboxes for 'Raised Exceptions', 'Unc caught Exceptions', and 'User Uncaught Except...', and a status bar at the bottom showing 'Ln 81, Col 14' and 'Python 3.11.5'.

```

File Edit Selection View ... ← → Search
RUN AND DEBUG RUN Welcome aac45.py
C:\> Users > boora > Downloads > aac45.py > travel_few_shot
69     prompt = f"""
70     Example 1:
71     Email: I was charged extra.
72     Category: Billing
73
74     Example 2:
75     Email: App crashes on login.
76     Category: Technical Support
77
78     Example 3:
79     Email: Great service!
80     Category: Feedback
81
82     Now classify:
83     Email: {query}
84     Category:
85     """
86     return llm_simulator(prompt)
87
88
89
90 # 2. TRAVEL QUERY CLASSIFICATION
91
92 def travel_few_shot(query):
93

```

This screenshot shows the continuation of the 'aac45.py' script. The 'travel_few_shot' function is now fully implemented, using the same pattern of examples and classification logic as the email functions. The code editor interface remains consistent with the first screenshot, including the 'RUN AND DEBUG' sidebar and the status bar at the bottom.

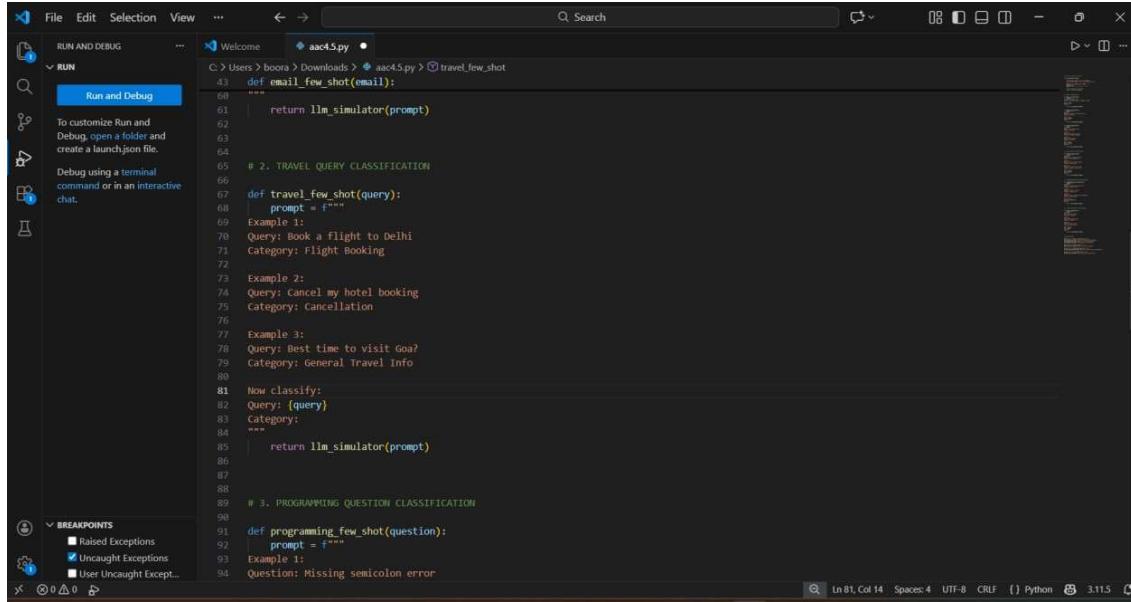
2. Travel Query Classification Scenario:

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

Tasks:

- Prepare labeled travel queries.
- Apply Zero-shot prompting.

- c. Apply One-shot prompting.
- d. Apply Few-shot prompting.
- e. Compare response consistency.



```

File Edit Selection View ... ← → Q Search RUN and DEBUG Welcome aac45.py ...
C:\> Users > boora > Downloads > aac45.py > travel_few_shot
43 def email_few_shot(email):
44     """
45         return llm_simulator(prompt)
46
47 # 2. TRAVEL QUERY CLASSIFICATION
48
49 def travel_few_shot(query):
50     prompt = f"""
51 Example 1:
52 Query: Book a flight to Delhi
53 Category: Flight Booking
54
55 Example 2:
56 Query: Cancel my hotel booking
57 Category: Cancellation
58
59 Example 3:
60 Query: Best time to visit goa?
61 Category: General Travel Info
62
63 Now classify:
64 Query: {query}
65 Category:
66
67         return llm_simulator(prompt)
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91 def programming_few_shot(question):
92     prompt = f"""
93 Example 1:
94 Question: Missing semicolon error

```

The screenshot shows a code editor window with a Python file named 'aac45.py'. The code defines three functions: 'email_few_shot', 'travel_few_shot', and 'programming_few_shot'. Each function takes a single parameter and returns a string generated by an 'llm_simulator'. The 'travel_few_shot' function includes examples for booking a flight to Delhi, canceling a hotel booking, and visiting Goa. The 'programming_few_shot' function includes an example of a question about a missing semicolon error. The code editor interface includes a 'RUN and DEBUG' sidebar, a 'BREAKPOINTS' section, and status bar information like 'Ln 81, Col 14' and 'Python 3.11.5'.

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.

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with 'RUN AND DEBUG' and 'BREAKPOINTS' sections. The main area displays a Python script named 'aac45.py'. The script contains several functions: 'travel_few_shot', 'programming_few_shot', 'social_few_shot', and 'llm_simulator'. The 'travel_few_shot' function includes examples for 'Example 1' (Missing semicolon error) and 'Example 2' (Output is wrong). The 'programming_few_shot' function includes 'Example 3' (How to reduce time complexity). The 'social_few_shot' function includes 'Example 1'. The code editor has a status bar at the bottom showing file information like 'Ln 81, Col 14' and encoding 'UTF-8'.

```
C:\>Users>boora>Downloads> aac45.py > travel_few_shot
67 def travel_few_shot(query):
68     """
69     To customize Run and
70     Debug, open a folder and
71     create a launch.json file.
72     Debug using a terminal
73     command or in an interactive
74     chat.
75     """
76     return llm_simulator(prompt)
77
78     # 3. PROGRAMMING QUESTION CLASSIFICATION
79
80     def programming_few_shot(question):
81         prompt = f"""
82             Example 1:
83             Question: Missing semicolon error
84             Category: Syntax Error
85
86             Example 2:
87             Question: output is wrong
88             Category: Logic Error
89
90             Example 3:
91             Question: How to reduce time complexity?
92             Category: Optimization
93
94             Now classify:
95             Question: {question}
96             Category:
97             """
98
99         return llm_simulator(prompt)
100
101
102     # 4. SOCIAL MEDIA POST CLASSIFICATION
103
104     def social_few_shot(post):
105         prompt = f"""
106             Example 1:
107             """
108
109
110
111
112
113
114
115
116
117
```

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.

The screenshot shows a code editor interface with a Python file named 'aac45.py' open. The code is a script for social media post classification, utilizing a function named 'social_few_shot' which takes a 'post' parameter and returns a 'Category'. It includes examples for promotion, complaint, appreciation, and a main execution section. The code editor has a sidebar with 'RUN AND DEBUG' and 'BREAKPOINTS' sections.

```
112
113 # 4. SOCIAL MEDIA POST CLASSIFICATION
114
115 def social_few_shot(post):
116     prompt = f"""
117     Example 1:
118     Post: Huge sale today!
119     Category: Promotion
120
121     Example 2:
122     Post: Worst service ever
123     Category: complaint
124
125     Example 3:
126     Post: love this brand
127     Category: Appreciation
128
129     Now classify:
130     Post: {post}
131     Category:
132     """
133     return llm_simulator(prompt)
134
135
136
137 # MAIN EXECUTION
138
139 print("\n===== EMAIL CLASSIFICATION =====")
140 print(email_zero_shot("I have not received my invoice"))
141 print(email_one_shot("Unable to reset my password"))
142 print(email_few_shot("The website is very slow"))
143
144 print("\n===== TRAVEL QUERY =====")
145 print(travel_few_shot("Cancel my flight ticket"))
146
147 print("\n===== PROGRAMMING QUESTION =====")
```

OUTPUT:

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\boora\Downloads> & 'c:\Users\boora\AppData\Local\Programs\Python\Python311-5.18.0-win32-x64\bundled\libs\debugpy\launcher' '50529' '--' 'C:\Users\boora\Downloads\email_classification.py'

===== EMAIL CLASSIFICATION =====

--- PROMPT SENT TO MODEL ---

Classify the email into:
Billing, Technical Support, Feedback, Others

Email: I have not received my invoice
Category:

-----
Predicted Category

--- PROMPT SENT TO MODEL ---

Example:
Email: My payment was deducted twice.
Category: Billing

Now classify:
Email: Unable to reset my password
Category:

-----
Predicted Category

--- PROMPT SENT TO MODEL ---

Example 1:
Email: I was charged extra.
Category: Billing

Example 2:
Email: App crashes on login.
Category: Technical Support

Example 3:
```

```
Example 1:  
Question: Missing semicolon error  
Category: Syntax Error
```

```
Example 2:  
Question: Output is wrong  
Category: Logic Error
```

```
Example 3:  
Question: How to reduce time complexity?  
Category: Optimization
```

```
Now classify:  
Question: Why is my loop running infinitely?  
Category:
```

Predicted Category

===== SOCIAL MEDIA POST =====

--- PROMPT SENT TO MODEL ---

```
Example 1:  
Post: Huge sale today!  
Category: Promotion
```

```
Example 2:  
Post: Worst service ever  
Category: Complaint
```

```
Example 2:  
Post: Love this brand  
Example 2:  
Post: Worst service ever  
Category: Complaint
```

```
Example 3:  
Post: Love this brand  
Category: Complaint
```

```
Example 3:  
Post: Love this brand
```

```
Example 3:  
Post: Love this brand
```