

Assignment – 4.5

2303A51641

Task 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.

PROMPT :

Classify the above below samples into one of the following categories: Billing, Technical Support, Feedback, Others.

Classify the above email samples into one of the following categories: Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month.'

Classify the below email samples into one of the following categories: Billing, Technical Support, Feedback, Others. Email: 'I have not received my invoice for last month. Billing, "Subject: Invoice #12345 Dear Customer, your invoice for the month of June is attached. Please make the payment by the due date. Best regards, Billing Team", "Technical Support", "Subject: Issue with Software Installation Hello Support Team, I am facing issues while installing the software on my computer. It shows an error code 404. Please assist. Thanks, User".'

CODE and OUTPUT :

```
AI-4.5.py > classify_multiple_emails
1 #Management wants to automatically classify emails into categories like "Billing", "Technical Support", "Feedback", and "Others" before assigning them to a
2 #Create or collect 10 short email samples, each belonging to one of the 4 categories.
3 #Classify the above below samples into one of the following categories: Billing, Technical Support, Feedback, Others.'"
4 def classify_email(email_content):
5     """Classify the email content into one of the categories: Billing, Technical Support, Feedback, Others.
6     Parameters:
7     email_content (str): The content of the email to classify.
8     Returns:
9     str: The category of the email."""
10    email_content = email_content.lower()
11
12    if any(keyword in email_content for keyword in ["invoice", "payment", "billing", "due date", "overdue"]):
13        return "Billing"
14    elif any(keyword in email_content for keyword in ["issue", "error", "support", "installation", "bug", "connectivity"]):
15        return "Technical Support"
16    elif any(keyword in email_content for keyword in ["feedback", "suggestion", "purchase", "quality", "service"]):
17        return "Feedback"
18    else:
19        return "Others"
20
21 email_samples = [
22     ("Billing", "Subject: Invoice #12345\nDear Customer, your invoice for the month of June is attached. Please make the payment by the due date.\nBest reg
23     ("Technical Support", "Subject: Issue with Software Installation\nHello Support Team, I am facing issues while installing the software on my computer.
24     ("Feedback", "Subject: Feedback on Recent Purchase\nHi Team, I recently purchased a product from your store and I am very satisfied with the quality ar
25     ("Others", "Subject: Meeting Reminder\nDear Team, this is a reminder for our meeting scheduled tomorrow at 10 AM in the conference room. Please be on t
26     ("Billing", "Subject: Payment Confirmation\nDear Customer, we have received your payment for the invoice #67890. Thank you for your prompt payment.\nBe
27     ("Technical Support", "Subject: Network Connectivity Issue\nHello, I am experiencing frequent disconnections from the internet. Can you please help me
28     ("Feedback", "Subject: Suggestion for New Features\nHi Team, I would like to suggest a few new features for your app that I believe would enhance user
29     ("Others", "Subject: Holiday Announcement\nDear All, please note that the office will be closed next Friday in observance of the holiday. Enjoy your d
30     ("Billing", "Subject: Overdue Payment Notice\nDear Customer, our records indicate that your payment for invoice #54321 is overdue. Please make the paym
31     ("Technical Support", "Subject: Software Bug Report\nHello Support, I have encountered a bug in the latest version of your software. It crashes when I
32 ]
33 print(classify_email(email_samples[1][1])) # Output: Technical Support
34 print(classify_email(email_samples[4][1])) # Output: Billing
35 print(classify_email(email_samples[7][1])) # Output: Others
36
37 #Classify the above email samples into one of the following categories: Billing, Technical Support, Feedback, Others.Email: 'I have not received my invoice
38 def classify_email_single(email_content):
39     """Classify a single email content into one of the categories: Billing, Technical Support, Feedback, Others.
40     Parameters:
```

```

AI-4.5.py > classify_multiple_emails
37 def classify_email_single(email_content):
38     Parameters:
39     email_content (str): The content of the email to classify.
40     Returns:
41     str: The category of the email.
42     email_content = email_content.lower()
43
44     if any(keyword in email_content for keyword in ["invoice", "payment", "billing", "due date", "overdue"]):
45         return "Billing"
46     elif any(keyword in email_content for keyword in ["issue", "error", "support", "installation", "bug", "connectivity"]):
47         return "Technical Support"
48     elif any(keyword in email_content for keyword in ["feedback", "suggestion", "purchase", "quality", "service"]):
49         return "Feedback"
50     else:
51         return "Others"
52
53 # Example Usage:
54 email_to_classify = "I have not received my invoice for last month."
55 print(classify_email_single(email_to_classify)) # Output: Billing
56
57 #Classify the below email samples into one of the following categories: Billing, Technical Support, Feedback, Others.Email: 'I have not received my invoice
58 def classify_multiple_emails(email_contents):
59     """Classify multiple email contents into one of the categories: Billing, Technical Support, Feedback, Others.
60     Parameters:
61     email_contents (list): A list of email contents to classify.
62     Returns:
63     list: A list of categories corresponding to each email.
64     categories = []
65     for email_content in email_contents:
66         email_content = email_content.lower()
67
68         if any(keyword in email_content for keyword in ["invoice", "payment", "billing", "due date", "overdue"]):
69             categories.append("Billing")
70         elif any(keyword in email_content for keyword in ["issue", "error", "support", "installation", "bug", "connectivity"]):
71             categories.append("Technical Support")
72         elif any(keyword in email_content for keyword in ["feedback", "suggestion", "purchase", "quality", "service"]):
73             categories.append("Feedback")
74         else:
75             categories.append("Others")
76     return categories
77
78 # Example Usage:
79 emails_to_classify = [
80     "I have not received my invoice for last month.",
81     "I am facing issues while installing the software on my computer. It shows an error code 404. Please assist."
82 ]
83 print(classify_multiple_emails(emails_to_classify)) # Output: ['Billing', 'Technical Support']
84

```

```

Technical Support
Billing
Others
Billing
['Billing', 'Technical Support']

```

Justification :

This task shows how prompt engineering can automate email sorting without building a custom model. Zero-shot prompting works for clear emails but may fail for overlapping categories. One-shot prompting improves understanding by providing a reference example. Few-shot prompting offers better accuracy by learning category patterns. This approach reduces manual effort and improves operational efficiency.

Task 2: Travel Query Classification

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.
- Apply One-shot prompting.
- Apply Few-shot prompting.
- Compare response consistency.

PROMPT :

A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info. Tasks: a. Prepare labeled travel queries.

Classify the travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New York next week.",

Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New York next week.", "Can you help me find a hotel in Paris?"

Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "Can you help me find a hotel in Paris?" "I would like to cancel my reservation for tomorrow.", "What are the travel restrictions for Italy?"

CODE and OUTPUT :

```
AI-4.5.py > classify_multiple_travel_queries
116 def classify_single_travel_query(query):
117     """
124     if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
125         return "Flight Booking"
126     elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
127         return "Hotel Booking"
128     elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
129         return "Cancellation"
130     else:
131         return "General Travel Info"
132
133 # Example Usage:
134 single_query = "Can you help me find a hotel in Paris?"
135 print(classify_single_travel_query(single_query)) # Output: Hotel Booking
136
137 #Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New York next week.", "Can you help me find a hotel in Paris?"
138 def classify_multiple_travel_queries(queries):
139     """Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
140     Parameters:
141     queries (list): A list of travel queries to classify.
142     Returns:
143     list: A list of categories corresponding to each travel query."""
144     categories = []
145     for query in queries:
146         query = query.lower()
147         if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
148             categories.append("Flight Booking")
149         elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
150             categories.append("Hotel Booking")
151         elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
152             categories.append("Cancellation")
153         else:
154             categories.append("General Travel Info")
155     return categories
156
157 # Example Usage:
158 queries_to_classify = [
159     "I need to book a flight to New York next week.",
160     "Can you help me find a hotel in Paris?"
161 ]
162 print(classify_multiple_travel_queries(queries_to_classify))
```

```

AI-4.5.py > classify_multiple_travel_queries
84
85 #A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info. Tasks: a. Prepare labeled travel queries
86 def classify_travel_query(query):
87     """Classify the travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
88     Parameters:
89     query (str): The travel query to classify.
90     Returns:
91     str: The category of the travel query."""
92     query = query.lower()
93
94     if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
95         return "Flight Booking"
96     elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
97         return "Hotel Booking"
98     elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
99         return "Cancellation"
100     else:
101         return "General Travel Info"
102
103 # Example Usage:
104 travel_queries = [
105     "I need to book a flight to New York next week.",
106     "Can you help me find a hotel in Paris?",
107     "I would like to cancel my reservation for tomorrow.",
108     "What are the travel restrictions for Italy?"
109 ]
110 for query in travel_queries:
111     print(f"Query: {query}\nCategory: {classify_travel_query(query)}\n")
112 # Output: # Query: I need to book a flight to New York next week.
113 # Category: Flight Booking
114
115 #Classify the travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "I need to book a flight to New York
116 def classify_single_travel_query(query):
117     """Classify a single travel query into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
118     Parameters:
119     query (str): The travel query to classify.
120     Returns:
121     str: The category of the travel query."""
122     query = query.lower()

```

```

#Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info. "Can you help me find a hotel
def classify_more_travel_queries(queries):
    """Classify multiple travel queries into one of the categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info.
    Parameters:
    queries (list): A list of travel queries to classify.
    Returns:
    list: A list of categories corresponding to each travel query."""
    categories = []
    for query in queries:
        query = query.lower()

        if any(keyword in query for keyword in ["flight", "airline", "ticket", "departure", "arrival"]):
            categories.append("Flight Booking")
        elif any(keyword in query for keyword in ["hotel", "accommodation", "room", "stay", "booking"]):
            categories.append("Hotel Booking")
        elif any(keyword in query for keyword in ["cancel", "cancellation", "refund", "reschedule"]):
            categories.append("Cancellation")
        else:
            categories.append("General Travel Info")
    return categories
# Example Usage:
more_queries_to_classify = [
    "Can you help me find a hotel in Paris?",
    "I would like to cancel my reservation for tomorrow.",
    "What are the travel restrictions for Italy?"
]
print(classify_more_travel_queries(more_queries_to_classify)) # Output: ['Hotel Booking', 'Cancellation', 'General Travel Info']

```

```

er' '55871' '--' 'D:\AI\AI-4.5.py'
Query: I need to book a flight to New York next week.
Category: Flight Booking

Query: Can you help me find a hotel in Paris?
Category: Hotel Booking

Query: I would like to cancel my reservation for tomorrow.
Category: Cancellation

Query: What are the travel restrictions for Italy?
Category: General Travel Info

Hotel Booking
['Flight Booking', 'Hotel Booking']
['Hotel Booking', 'Cancellation', 'General Travel Info']
PS D:\AI>

```

Justification :

Travel-related queries often use similar language for different intents. Zero-shot prompting may misclassify such queries due to lack of context. One-shot prompting helps by giving a basic example. Few-shot prompting improves consistency and intent recognition. This task highlights the importance of contextual examples in user-facing assistants.

Task 3. Programming Question Type Identification

Scenario:

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

Tasks:

- Prepare coding-related user queries.
- Perform Zero-shot classification.
- Perform One-shot classification.
- Perform Few-shot classification.
- Analyze improvements in technical accuracy.

PROMPT :

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.

CODE and OUTPUT :

```
191 #A coding help chatbot must classify queries into Syntax Error, Logic Error, Optimization, or Conceptual Question. Tasks:a. Prepare coding-related user qu
192 def classify_coding_query(query):
193     """Classify the coding query into one of the categories: Syntax Error, Logic Error, Optimization, Conceptual Question.
194     Parameters:
195     query (str): The coding query to classify.
196     Returns:
197     str: The category of the coding query."""
198     query = query.lower()
199
200     if any(keyword in query for keyword in ["syntax error", "unexpected indent", "missing parenthesis", "invalid syntax"]):
201         return "Syntax Error"
202     elif any(keyword in query for keyword in ["logic error", "wrong output", "incorrect result", "bug"]):
203         return "Logic Error"
204     elif any(keyword in query for keyword in ["optimize", "performance", "efficiency", "speed up"]):
205         return "Optimization"
206     elif any(keyword in query for keyword in ["how to", "what is", "explain", "concept"]):
207         return "Conceptual Question"
208     else:
209         return "Others"
210
211 # Example Usage:
212 coding_queries = [
213     "I am getting a syntax error when I run my Python code.",
214     "My program is producing the wrong output, what could be the logic error?",
215     "How can I optimize my code for better performance?",
216     "Can you explain the concept of recursion in programming?"
217 ]
218 for query in coding_queries:
219     print(f"Query: {query}\nCategory: {classify_coding_query(query)}\n")
220 # Output:
221 # Query: I am getting a syntax error when I run my Python code.
222 # Category: Syntax Error
223 # Query: My program is producing the wrong output, what could be the logic error?
224 # Category: Logic Error
225 # Query: How can I optimize my code for better performance?
226 # Category: Optimization
227 # Query: Can you explain the concept of recursion in programming?
228 # Category: Conceptual Question
```

```
C:\PS> python D:\AI\AI-4.5.py
```

```
Query: I am getting a syntax error when I run my Python code.
Category: Syntax Error
```

```
Query: My program is producing the wrong output, what could be the logic error?
Category: Logic Error
```

```
Query: How can I optimize my code for better performance?
Category: Optimization
```

```
Query: Can you explain the concept of recursion in programming?
Category: Conceptual Question
```

```
C:\PS D:\AI>
```

Justification :

Programming queries require technical and contextual understanding. Zero-shot prompting may confuse syntax and logic-related issues. One-shot prompting provides some guidance to the model. Few-shot prompting significantly improves technical accuracy. This demonstrates how examples enhance domain-specific classification.

Task 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.

PROMPT :

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

CODE and OUTPUT :

```
229 #Social Media Post Categorization
230 # Scenario:
231 # A social media analytics tool must classify posts into Promotion,
232 # Complaint, Appreciation, or Inquiry.
233 # Tasks:
234 # 1. Prepare sample social media posts.
235 # 2. Use Zero-shot prompting.
236 # 3. Use One-shot prompting.
237 # 4. Use Few-shot prompting.
238 # 5. Analyze informal language handling.
239 def classify_social_media_post(post):
240     """Classify the social media post into one of the categories: Promotion, Complaint, Appreciation, Inquiry.
241     Parameters:
242     post (str): The social media post to classify.
243     Returns:
244     str: The category of the social media post."""
245     post = post.lower()
246
247     if any(keyword in post for keyword in ["buy now", "sale", "discount", "offer", "promo"]):
248         return "Promotion"
249     elif any(keyword in post for keyword in ["not happy", "disappointed", "bad service", "complaint", "issue"]):
250         return "Complaint"
251     elif any(keyword in post for keyword in ["thank you", "great job", "love it", "appreciate", "awesome"]):
252         return "Appreciation"
253     elif any(keyword in post for keyword in ["how to", "where can i", "what is", "help me"]):
254         return "Inquiry"
255     else:
256         return "Others"
257 # Example Usage:
258 social_media_posts = [
259     "Huge sale on all products! Buy now and save big!",
260     "I'm really disappointed with the service I received today.",
261     "Thank you for the amazing support! You guys are awesome!",
262     "Can someone help me with my account settings?",
263     "Just wanted to share how much I love this new app!"
264 ]
265 for post in social_media_posts:
266     print(f"Post: {post}\nCategory: {classify_social_media_post(post)}\n")
```

```

AI-4.5.py > ...
260     "I'm really disappointed with the service I received today.",
261     "Thank you for the amazing support! You guys are awesome!",
262     "Can someone help me with my account settings?",
263     "Just wanted to share how much I love this new app!"
264 ]
265 for post in social_media_posts:
266     print(f"Post: {post}\nCategory: {classify_social_media_post(post)}\n")
267 # Output:
268 # Post: Huge sale on all products! Buy now and save big!
269 # Category: Promotion
270 # Post: I'm really disappointed with the service I received today.
271 # Category: Complaint
272 # Post: Thank you for the amazing support! You guys are awesome!
273 # Category: Appreciation
274 # Post: Can someone help me with my account settings?
275 # Category: Inquiry
276 # Post: Just wanted to share how much I love this new app!
277 # Category: Appreciation
278

```

```

● Post: Huge sale on all products! Buy now and save big!
  Category: Promotion

Post: I'm really disappointed with the service I received today.
Category: Complaint

Post: Thank you for the amazing support! You guys are awesome!
Category: Appreciation

Post: Can someone help me with my account settings?
Category: Inquiry

Post: Just wanted to share how much I love this new app!
Category: Others

```

Justification :

Social media posts often contain informal language, slang, and emojis. Zero-shot prompting may struggle to interpret tone and intent correctly. One-shot prompting slightly improves classification. Few-shot prompting handles informal expressions more effectively. This task proves the value of examples in understanding unstructured text.