

ASSIGNMENT-4.5

NAME-ARSHA VARDHINI

BATCH-29

ROLL-NO:2303A51600

ADVANCED PROMPT ENGINEERING: ZERO-SHOT, ONE-SHOT & FEW-SHOT

TASK-1:

ZERO-SHOT

A. Preparing Sample data:

```
test_emails = [  
    "My payment failed but money was deducted.",  
    "The app is not opening on my phone.",  
    "Great customer service, very satisfied.",  
    "What is your customer care number?",  
    "Invoice amount seems incorrect."  
]
```

Expected Labels (for evaluation):

```
true_labels = [  
    "Billing",  
    "Technical Support",
```

"Feedback",

"Others",

"Billing"

]

PROMPT:

Classify the following email into one of the categories:

Billing, Technical Support, Feedback, Others.

Email: "<email_text>"

Return only the category name.

CODE:

The screenshot shows a code editor interface with multiple tabs at the top. The active tab is 'Assignment-4.5.py'. The code in the editor is as follows:

```
1 """Classify the following email into one of the categories:
2 Billing, Technical Support, Feedback, Others.
3
4 Email: "<email_text>"
5 Return only the category name.
6 """
7 #sample data
8 test_emails = [
9     "My payment failed but money was deducted.",
10    "The app is not opening on my phone.",
11    "Great customer service, very satisfied.",
12    "What is your customer care number?",
13    "Invoice amount seems incorrect."
]
```

Below the code editor, there is a terminal window showing the following interaction:

```
Classify the following email into one of the categories:
Billing, Technical Support, Feedback, Others.

Email: "Invoice amount seems incorrect."

AI Output: Others
```

The terminal also shows the path 'PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>' and the Python version '3.14.2'.

OBSERVATION:

Classifies emails using only instructions, without examples.
Works if keywords are clear, may misclassify ambiguous emails.
Quick and simple, but less accurate for complex cases.

ONE-SHOT :

PROMPT: Example:

Email: "I was charged twice for my last payment."

Category: Billing

Now classify the following email:

Email: "<email_text>"

CODE:

The screenshot shows a code editor interface with several tabs at the top: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, and Assignment-4.5.py (which is the active tab). The code in Assignment-4.5.py is as follows:

```
ASS-1.5.py
As-2.5.py
Assignment-3.1.py
Assignment-4.2.py
Untitled-1
Assignment-4.5.py X

Assignment-4.5.py > ...
22 def one_shot_classification(emails):
23     print("\n===== ONE-SHOT PROMPTING =====\n")
24     predictions = []
25
26     for email in emails:
27         print("Prompt sent to AI:")
28         print(f"""
29 Example:
30 Email: "I was charged twice for my last payment."
31 Category: Billing
32
33 Now classify the following email:
34
35 Email: "{email}"
```

The terminal below shows the execution of the script:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Category: Billing
Now classify the following email:
Email: "Invoice amount seems incorrect."
AI Output: Billing
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

The status bar at the bottom indicates the file is in Python mode, has 3.14.2 syntax, and is using Python 3.14.2.

OBSERVATION:

Provides one example to guide the AI's reasoning.

Improves accuracy over zero-shot and handles slightly ambiguous emails better.

Still limited; accuracy depends on how representative the example is.

One example helps the AI understand the expected format and category mapping.

Classification accuracy improves compared to zero-shot, especially for similar issues.

Performance depends heavily on how relevant the single example is to the new email.

FEW SHOT:

PROMPT:

Email: "I was charged twice for my last payment." → Billing

Email: "The app crashes on login." → Technical Support

Email: "I love the new update." → Feedback

Email: "What are your office hours?" → Others

CODE:

The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the following details:

- File Menu:** File, Edit, Selection, View, ...
- Search Bar:** Q AI-Assitant coding
- Toolbar:** Back, Forward, Home, Minimize, Maximize, Close
- Left Sidebar:** Includes icons for File, Find, Replace, Go To, Open, Save, and a Help section.
- Tab Bar:** ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py (active tab).
- Code Editor:** The active file, Assignment-4.5.py, contains Python code for few-shot classification. It includes examples of emails and their categories.

```
22 def few_shot_classification(emails):
23     print("\n===== FEW-SHOT PROMPTING =====\n")
24     predictions = []
25
26     for email in emails:
27         print("Prompt sent to AI:")
28         print(f"""
29 Examples:
30 Email: "I was charged twice for my last payment."
31 Category: Billing
32
33 Email: "The app crashes on login."
34 Category: Technical Support
35
```

- Bottom Navigation:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), PORTS.
- Terminal:** Displays the command PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding> [redacted]
- Right Sidebar:** A sidebar titled "Python Environments" lists multiple Python environments: Python, Python, Python, powershell, Python, powershell, and Python.

OBSERVATION:

Provides multiple examples to show patterns to the AI.

Highest accuracy; AI can generalize better for unseen emails.

Slightly longer prompts but most reliable for real-world use

TASK-2:

```
# Sample travel queries (short & simple)
```

```
travel_queries = [
```

```
    "Book a flight from Delhi to Mumbai.",
```

```
    "Cancel my hotel reservation in Paris.",
```

```
    "What is the baggage allowance?",
```

```
    "I need a hotel in London for 2 nights.",
```

```
    "Cancel my flight ticket to New York."
```

```
]
```

```
# True labels for evaluation
```

```
true_labels = [
```

```
    "Flight Booking",
```

```
    "Cancellation",
```

```
    "General Travel Info",
```

```
    "Hotel Booking",
```

```
    "Cancellation"
```

```
]
```

ZERO-SHOT:

PROMPT: Classify the following travel query into one of the categories:

Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: "<travel_query>"

CODE:

The screenshot shows a Python script named `Assignment-4.5.py` in a code editor. The code defines a list of true labels and a function for zero-shot travel classification. The output terminal shows AI responses for hotel booking and cancellation queries.

```
10 # True labels for evaluation
11 true_labels = [
12     "Flight Booking",
13     "Cancellation",
14     "General Travel Info",
15     "Hotel Booking",
16     "Cancellation"
17 ]
18 def zero_shot_travel_classification(queries):
19     print("\n==== ZERO-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI:\nQuery: \"{query}\"\n")
24
25         # AI processing logic (omitted)
26
27         if query in true_labels:
28             prediction = "True"
29         else:
30             prediction = "False"
31
32         predictions.append(prediction)
33
34     return predictions
35
36
37 # Test cases
38 queries = ["Cancel my flight ticket to New York.", "Book a flight from New York to London next week"]
39 predictions = zero_shot_travel_classification(queries)
40
41 for i, query in enumerate(queries):
42     print(f"AI Output: {predictions[i]}")
43     print("-----")
44
45     if predictions[i] == "True":
46         print(f"Prompt sent to AI:\nQuery: \"{query}\"\n")
47
48         # AI processing logic (omitted)
49
50         if query in true_labels:
51             print("AI Response: Hotel Booking")
52         else:
53             print("AI Response: Cancellation")
54
55         print("-----")
56
57     else:
58         print("AI Response: None")
59
60         print("-----")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

AI Output: Hotel Booking

Prompt sent to AI:
Query: "Cancel my flight ticket to New York."

AI Output: Cancellation

PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assistent coding> []

OBSERVATION:

Classifies queries using only instructions, without examples.

Works for obvious keywords like “flight” or “cancel”, may misclassify tricky queries.

Fast and simple, but accuracy is lower for ambiguous cases.

ONE-SHOT:

PROMPT: Example:

Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel_query>"

CODE:

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, ..., <-, >, Q AI-Assitant coding.
- Tab Bar:** ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py (highlighted).
- Code Area:** Python code for "one_shot_travel_classification".

```
18 def one_shot_travel_classification(queries):
19     print("\n==== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with one example:\nQuery: \"{query}\"\\n")
24
25         if "cancel" in query.lower():
26             output = "Cancellation"
27         elif "flight" in query.lower():
28             output = "Flight Booking"
29         elif "hotel" in query.lower():
30             output = "Hotel Booking"
31         else:
```
- Terminal Output:**

```
AI Output: Hotel Booking
-----
Prompt sent to AI with one example:
Query: "Cancel my flight ticket to New York."
AI Output: Cancellation
```
- Bottom Bar:** PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (highlighted), PORTS.
- Right Sidebar:** A sidebar titled "TERMINAL" showing multiple terminal sessions:
 - + Python
 - + Python
 - + Python
 - + powershell
 - + Python
 - + powershell
 - + Python

OBSERVATION:

Provides one example to guide AI's reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the example is.

FEW SHOT:

PROMPT:

Examples:

Query: "Book a flight to Mumbai."

Category: Flight Booking

Query: "Cancel my hotel reservation."

Category: Cancellation

Query: "I need a hotel in London."

Category: Hotel Booking

Query: "What is the baggage allowance?"

Category: General Travel Info

Now classify the following query:

Query: "<travel_query>"

CODE:

The screenshot shows a code editor interface with multiple tabs at the top: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, and Assignment-4.5.py (which is the active tab). The code in Assignment-4.5.py is as follows:

```
18 def few_shot_travel_classification(queries):
19     print("\n==== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"\n")
24
25         if "cancel" in query.lower():
26             output = "Cancellation"
27         elif "flight" in query.lower():
28             output = "Flight Booking"
29         elif "hotel" in query.lower():
30             output = "Hotel Booking"
31         else:
```

The terminal below the code shows the AI's responses to different queries:

```
AI Output: Hotel Booking
-----
Prompt sent to AI with multiple examples:
Query: "Cancel my flight ticket to New York."
AI Output: Cancellation
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding> [
```

A sidebar on the right lists several Python files and PowerShell scripts.

OBSERVATION:

Provides multiple examples to show patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for real-world use.

TASK-3:

SAMPLE DATA:

```
# Sample coding queries (short & simple)
coding_queries = [
    "Why am I getting IndexError in my Python list?",
    "My sorting algorithm is too slow for large inputs.",
    "I wrote a function but it returns wrong results.",
    "Explain the difference between list and tuple in Python.",
    "How can I optimize my recursive Fibonacci function?"
]
```

True labels for evaluation

```
true_labels = [
    "Syntax Error",
    "Optimization",
    "Logic Error",
    "Conceptual Question",
    "Optimization"
]
```

ZERO-SHOT

PROMPT: Classify the following coding query into one of the categories:

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "<coding_query>"

CODE:

```
File Edit Selection View ... ← → ⌂ Ai-Assitant coding ⌂ Assignment-4.5.py X - ⌂ Assignment-4.5.py > ... ASS-1.5.py As-2.5.py Assignment-3.1.py Assignment-4.2.py Untitled-1 Assignment-4.5.py ⌂ Assignment-4.5.py > ... 18 def zero_shot_coding_classification(queries): 19     print("\n==== ZERO-SHOT PROMPTING ===\n") 20     predictions = [] 21 22     for query in queries: 23         print(f"Prompt sent to AI:\nQuery: \"{query}\"\n") 24 25         # Simulated AI based on keywords 26         if "error" in query.lower() or "IndexError" in query: 27             output = "Syntax Error" 28         elif "wrong" in query.lower(): 29             output = "Logic Error" 30         elif "slow" in query.lower() or "optimize" in query.lower(): 31             output = "Optimization" PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + ... ⌂ Python ⌂ Python ⌂ Python ⌂ powershell ⌂ Python ⌂ powershell ⌂ Python AI Output: Conceptual Question ----- Prompt sent to AI: Query: "How can I optimize my recursive Fibonacci function?" AI Output: Optimization ----- PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding> ⌂
```

OBSERVATION:

ONE SHOT

PROMPT:

Example:

Query: "I want to cancel my Python function."

Category: Logic Error

Now classify the following coding query:

Query: "<coding_query>"

CODE:

The screenshot shows a code editor interface with the following details:

- File Bar:** File, Edit, Selection, View, ..., <-, >, Q: AI-Assitant coding.
- Tab Bar:** ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py (highlighted).
- Code Area:** A Python script named Assignment-4.5.py. The code defines a function `one_shot_coding_classification` that takes a list of queries. It prints a prompt message and then iterates through the queries, classifying them based on specific keywords. The output categories include Syntax Error, Logic Error, Optimization, and Conceptual Question.
- Terminal Area:** Shows the command PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding> followed by a blank line.
- Right Sidebar:** A sidebar titled "Recent" showing a list of recent sessions: Python, powershell, Python, Python, powershell, Python, Python.

OBSERVATION:

Provides one example to guide AI's reasoning.

Improves accuracy and handles slightly ambiguous queries better.

Accuracy depends on how representative the single example is.

FEW SHOT

PROMPT:

Examples:

Query: "Why does my Python list give IndexError?"

Category: Syntax Error

Query: "My function returns wrong output."

Category: Logic Error

Query: "My loop is too slow for large data."

Category: Optimization

Query: "Explain Python variable scopes."

Category: Conceptual Question

Now classify the following coding query:

Query: "<coding_query>"

CODE:

The screenshot shows a code editor interface with a dark theme. The main window displays a Python file named 'Assignment-4.5.py' containing the following code:

```
18 def few_shot_coding_classification(queries):
19     print("\n==== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for query in queries:
23         print(f"Prompt sent to AI with multiple examples:\nQuery: \"{query}\"\n")
24
25         if "error" in query.lower() or "IndexError" in query:
26             output = "Syntax Error"
27         elif "wrong" in query.lower():
28             output = "Logic Error"
29         elif "slow" in query.lower() or "optimize" in query.lower():
30             output = "Optimization"
31         else:
```

Below the code editor, the terminal pane shows the AI's output:

```
AI Output: Conceptual Question
-----
Prompt sent to AI with multiple examples:
Query: "How can I optimize my recursive Fibonacci function?"
AI Output: Optimization
-----
```

The terminal also shows the current working directory as 'PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>'.

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; AI generalizes better for unseen queries.

Slightly longer prompts but most reliable for technical classification.

TASK-4

ZERO-SHOT

PROMPT:

Classify the following social media post into one of the categories:

Promotion, Complaint, Appreciation, Inquiry.

Post: "<social_post>"

CODE:

The screenshot shows a code editor interface with several tabs at the top: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, and Assignment-4.5.py (which is the active tab). The code in Assignment-4.5.py is as follows:

```
ASS-1.5.py
As-2.5.py
Assignment-3.1.py
Assignment-4.2.py
Untitled-1
Assignment-4.5.py

def zero_shot_social_classification(posts):
    print("\n==== ZERO-SHOT PROMPTING ===\n")
    predictions = []
    for post in posts:
        print(f"Prompt sent to AI:\nPost: '{post}'\n")
        # AI decides based on keywords
        if "disappointed" in post.lower() or "not arrived" in post.lower():
            output = "Complaint"
        elif "discount" in post.lower() or "offer" in post.lower():
            output = "Promotion"
        else:
            output = "Appreciation"
        predictions.append(output)
    return predictions
```

The terminal pane below shows the execution of the code and the AI's classification of a post:

```
AI Output: Inquiry
-----
Prompt sent to AI:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

A sidebar on the right lists multiple Python and PowerShell files.

OBSERVATION:

Classifies posts using only instructions, without examples.
Works for clear keywords but may misinterpret informal or slang language.

Fast and simple, lower accuracy for ambiguous or sarcastic posts.

ONE-SHOT

PROMPT:

Example:

Post: "My order is late."

Category: Complaint

Now classify the following social media post:

Post: "<social_post>"

CODE:

The screenshot shows a code editor window titled "AI-Assitant coding". The main pane displays a Python script named "Assignment-4.5.py". The code defines a function `one_shot_social_classification(posts)` that prints a message and then iterates through a list of posts. It checks if words like "disappointed" or "not arrived" are present in the post's lowercase version and sets the output to "Complaint". If words like "discount" or "offer" are present, it sets the output to "Promotion". The terminal pane at the bottom shows the execution of the script. It first prints "AI Output: Promotion". Then, it processes a post: "Can someone help me with the login issue?", which is classified as "Inquiry". Finally, it processes another post: "Thanks for the quick customer support!", which is classified as "Appreciation". The status bar at the bottom indicates the path "C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>".

```
ASS-7.5.py As-2.5.py Assignment-3.1.py Assignment-4.2.py Untitled-1 Assignment-4.5.py
Assignment-4.5.py > ...
18 def one_shot_social_classification(posts):
19     print("\n==== ONE-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI with one example:\nPost: \'{post}\'\n")
24
25         if "disappointed" in post.lower() or "not arrived" in post.lower():
26             output = "Complaint"
27         elif "discount" in post.lower() or "offer" in post.lower():
28             output = "Promotion"
...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
AI Output: Promotion
-----
Prompt sent to AI with one example:
Post: "Can someone help me with the login issue?"
AI Output: Inquiry
-----
Prompt sent to AI with one example:
Post: "Thanks for the quick customer support!"
AI Output: Appreciation
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

OBSERVATION:

Provides one example to guide AI reasoning.

Improves accuracy and handles some informal expressions

better.

Depends on how representative the example is for informal language.

FEW-SHOT

PROMPT:

Examples:

Post: "Loved the new feature!" → Appreciation

Post: "My order hasn't arrived." → Complaint

Post: "Check out our discount offer!" → Promotion

Post: "How can I reset my password?" → Inquiry

Now classify the following social media post:

Post: "<social_post>"

CODE:

The screenshot shows a code editor interface with multiple tabs at the top, including 'ASS-1.5.py', 'As-2.5.py', 'Assignment-3.1.py', 'Assignment-4.2.py', 'Untitled-1', and 'Assignment-4.5.py'. The 'Assignment-4.5.py' tab is active. The code in the editor is as follows:

```
ASS-1.5.py
As-2.5.py
Assignment-3.1.py
Assignment-4.2.py
Untitled-1
Assignment-4.5.py

Assignment-4.5.py > ...
def few_shot_social_classification(posts):
    print("\n==== FEW-SHOT PROMPTING ===\n")
    predictions = []
    for post in posts:
        print(f"Prompt sent to AI with multiple examples:\nPost: \"{post}\"\n")
        if "disappointed" in post.lower() or "not arrived" in post.lower():
            output = "Complaint"
        elif "discount" in post.lower() or "offer" in post.lower():
            output = "Promotion"
        else:
            output = "Inquiry"
        predictions.append(output)
    return predictions
```

The terminal pane below the code editor shows the execution of the script:

```
AI Output: Promotion
-----
Prompt sent to AI with multiple examples:
Post: "Can someone help me with the login issue?"

AI Output: Inquiry
-----
Prompt sent to AI with multiple examples:
Post: "Thanks for the quick customer support!"

AI Output: Appreciation
-----
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

A sidebar on the right lists several open files, mostly in Python and PowerShell.

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; better handles informal, slang, or mixed-language posts.

Slightly longer prompts but most reliable for social media classification.