

## **ASSIGNMENT-4.5**

**NAME-TEENESWARI**

**BATCH-29**

**ROLL-NO:2303A51932**

### **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 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:

```
"""Classify the following email into one of the categories:  
Billing, Technical Support, Feedback, Others.  
  
Email: <email_text>  
Return only the category name.  
"""\n#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."  
]  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

In the terminal panel below, the user has entered:

```
Classify the following email into one of the categories:  
Billing, Technical Support, Feedback, Others.  
  
Email: "Invoice amount seems incorrect."  
  
AI Output: Others
```

The status bar at the bottom shows the path PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>, the Python version 3.14.2, and the date/time 10:00 AM 23-01-2026.

## 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:

```
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 AI's response:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Category: Billing
Now classify the following email:
Email: "Invoice amount seems incorrect."
AI Output: Billing
```

The status bar at the bottom indicates the file path PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding, line 54, column 1, and other settings like spaces:4, UTF-8, CRLF, Python, 3.142, and a few icons.

## 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:

A screenshot of the Visual Studio Code (VS Code) interface. The title bar says "AI-Assitant coding". The left sidebar shows file navigation with icons for file, folder, search, and others. The main editor tab is "Assignment-4.5.py". The code in the editor is:

```
Assignment-4.5.py > ...
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
```

The terminal tab is active, showing:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Category: Others
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 shows "Ln 63, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.2" and a yellow status icon.

## 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"
]
```

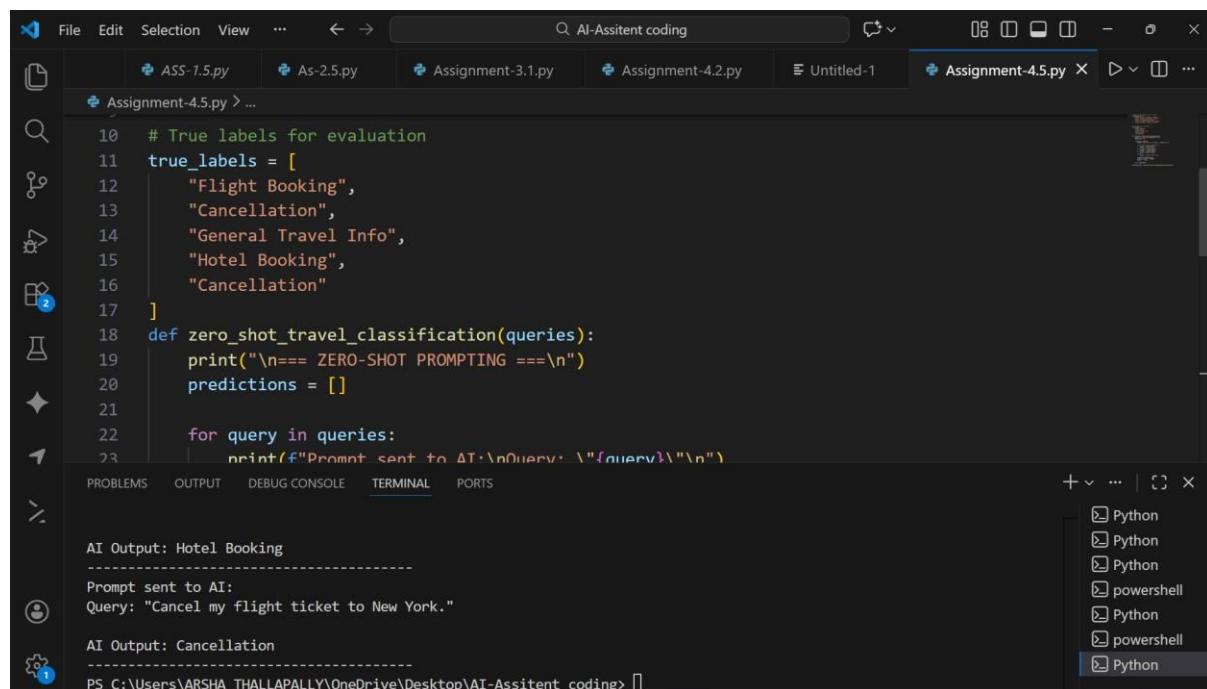
## ZERO-SHOT:

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 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:

```
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")
```

The terminal below the code shows the output of the script:

```
AI Output: Hotel Booking
-----
Prompt sent to AI:
Query: "Cancel my flight ticket to New York."
AI Output: Cancellation
-----
```

The bottom status bar shows the path 'PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>' and indicates there is 1 unread message.

## 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:

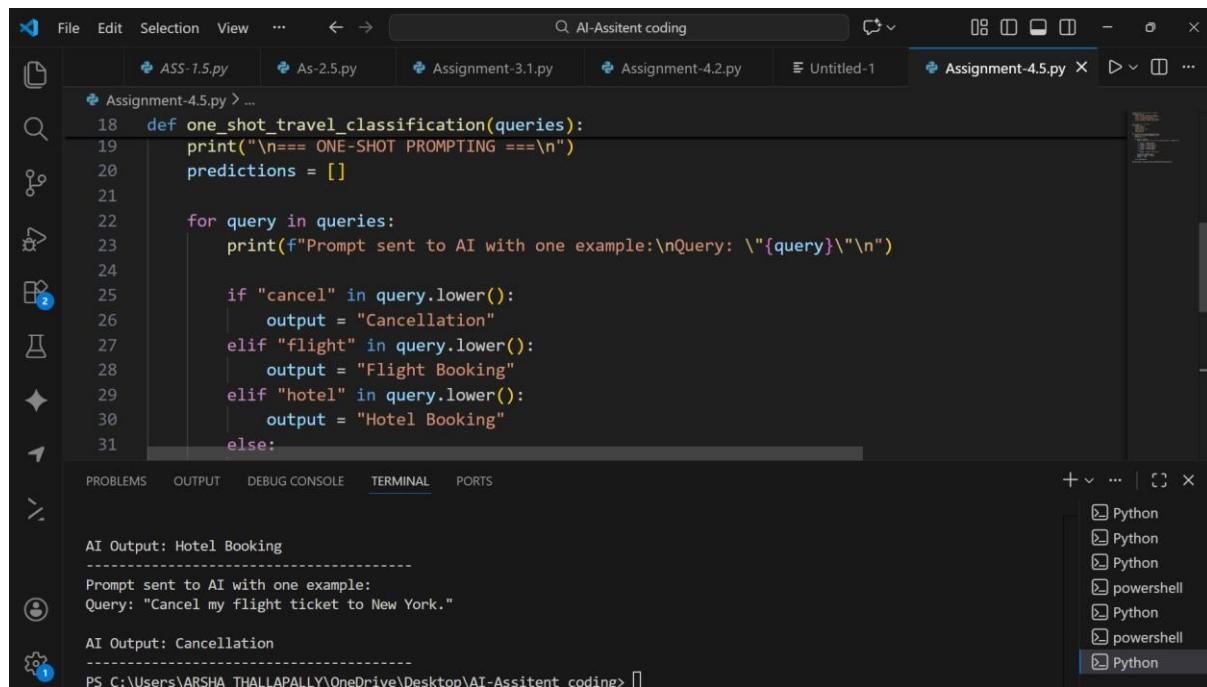
Example:

Query: "Cancel my flight ticket."

Category: Cancellation

Now classify the following query:

Query: "<travel\_query>" CODE:



The screenshot shows a code editor interface with a dark theme. The top menu bar includes File, Edit, Selection, View, and others. A search bar at the top right contains the text "AI-Assitant coding". Below the menu is a tab bar with several tabs: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py (which is the active tab), and Assignment-4.5.py. On the left side, there are various icons for file operations like copy, paste, and delete. The main code area displays the following Python script:

```
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:
```

Below the code, the terminal pane shows the output of the script. It starts with "AI Output: Hotel Booking" followed by a separator line. Then it says "Prompt sent to AI with one example:" and shows the query "Query: \"Cancel my flight ticket to New York.\"". Finally, it outputs "AI Output: Cancellation". At the bottom, the terminal shows the path "PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding> []".

## 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 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 defines a function `few_shot_travel_classification` that prints a few-shot prompting message and then classifies queries based on keywords like "cancel", "flight", or "hotel". The terminal below shows the AI's output for different queries, demonstrating its ability to generalize from examples. A sidebar on the right lists various ports and their connections.

```
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:
```

## 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

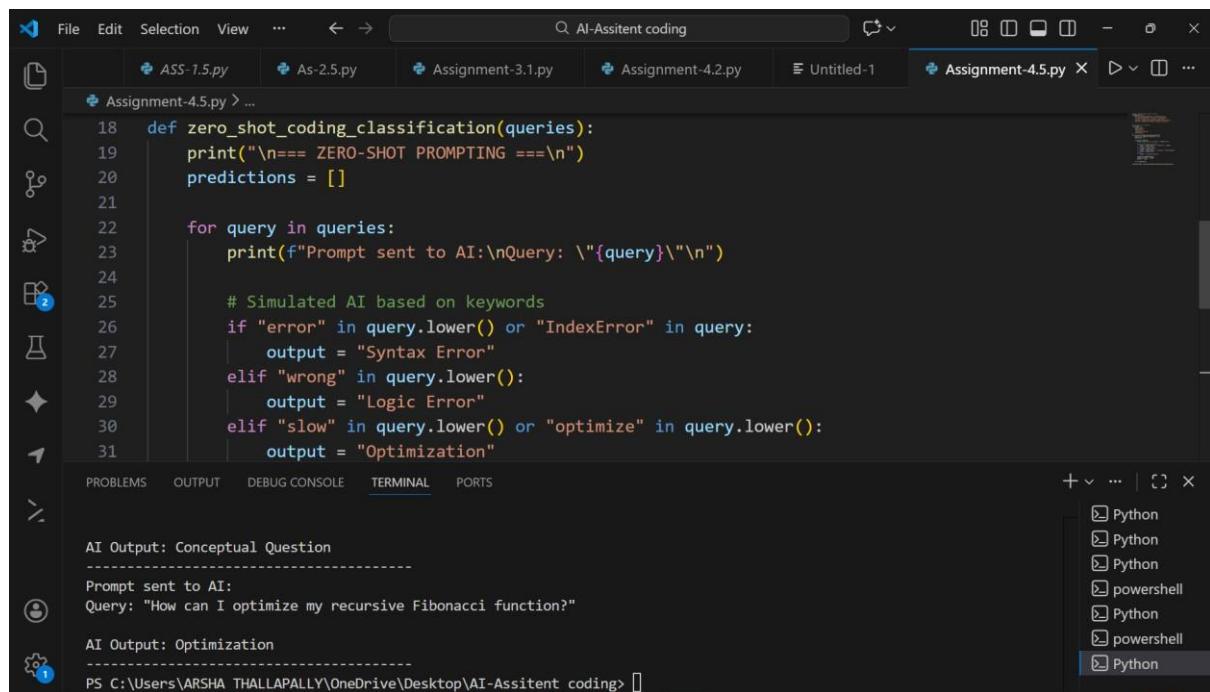
true_labels = [
    "Syntax
Error",
    "Optimization",
    "Logic Error",
    "Conceptual Question",
    "Optimization"
]
```

## ZERO-SHOT

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

Syntax Error, Logic Error, Optimization, Conceptual Question.

Query: "<coding\_query>" CODE:



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, and Assignment-4.5.py.
- Terminal:**

```
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>
```
- Bottom Right:** A sidebar titled "Python" lists multiple Python environments.

## 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 a dark theme. The top menu bar includes File, Edit, Selection, View, and other standard options. A search bar at the top right contains the text "AI-Assitant coding". Below the menu is a tab bar with several tabs: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py (which is currently selected), and a new tab icon. On the left side, there are various icons for file operations like Open, Save, Find, and Copy/Paste. The main code editor area contains the following Python code:

```
18 def one_shot_coding_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 "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, there is a terminal window showing AI output. The output starts with "AI Output: Conceptual Question" followed by a dashed line. It then says "Prompt sent to AI with one example:" and shows a sample query: "Query: How can I optimize my recursive Fibonacci function?". Another dashed line follows, and the output continues with "AI Output: Optimization". At the bottom of the terminal window, it shows the command prompt "PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>". To the right of the code editor, there is a sidebar with a list of open files, each with a small thumbnail preview: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py, and a new file icon.

## 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 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 main area displays a Python function named `few_shot_coding_classification` that prints a message and processes a list of queries. The code editor has a dark theme with light-colored syntax highlighting. Below the code editor is a terminal window showing AI-generated outputs for different prompts. The terminal tabs include PROBLEMS, OUTPUT, DEBUG CONSOLE (which is selected), and TERMINAL. The right sidebar shows a list of open terminals, all labeled "Python".

```
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:
```

AI Output: Conceptual Question  
-----  
Prompt sent to AI with multiple examples:  
Query: "How can I optimize my recursive Fibonacci function?"  
AI Output: Optimization  
-----  
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 main area contains Python code for a function named zero\_shot\_social\_classification. The code prints a prompt message and then iterates through a list of posts, checking for keywords like 'disappointed' or 'not arrived' to categorize them as 'Complaint'. The terminal pane below shows AI interactions:

```
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>
```

## 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 interface with a dark theme. At the top, there are tabs for various files: ASS-1.5.py, As-2.5.py, Assignment-3.1.py, Assignment-4.2.py, Untitled-1, Assignment-4.5.py (which is the active tab), and Assignment-4.5.py > ... . Below the tabs, the main code area contains the following Python code:

```
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"
```

Below the code, there are several sections labeled "AI Output" showing the classification results for different posts:

- 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

At the bottom, the file path is shown as 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:

A screenshot of the Visual Studio Code (VS Code) interface. The top bar shows the menu: File, Edit, Selection, View, ..., and a search bar with the text "AI-Assitant coding". Below the menu is a tab bar with several files: 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 currently active). The main editor area contains Python code for a few-shot social classification task:

```
Assignment-4.5.py > ...
18 def few_shot_social_classification(posts):
19     print("\n==== FEW-SHOT PROMPTING ===\n")
20     predictions = []
21
22     for post in posts:
23         print(f"Prompt sent to AI with multiple examples:\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"
```

The bottom left panel shows the "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS" tabs. The "OUTPUT" tab displays AI responses:

```
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
-----
```

The bottom right panel shows a list of open terminals:

- Python
- Python
- Python
- powershell
- Python
- powershell
- Python

The status bar at the bottom shows the path "C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>".

## 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.