Ai Assisted Coding

Week:4.5

Ht.No:2303A52350

Batch:45

Task:01. 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.

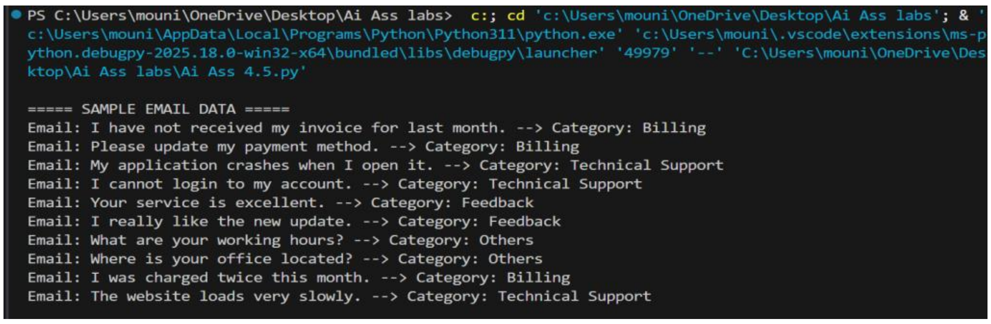
Prompt:

Create or collect 10 short email samples, each belonging to one of the 4 categories. Design a prompt that asks the LLM to classify a single email without providing any examples. “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.’”.Add new email Use 3–5 labeled examples in your prompt before asking the model to classify a new email.

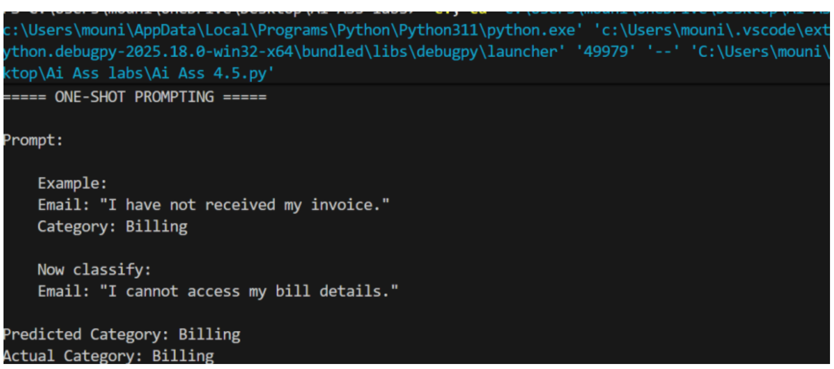
Code:

```
# Email Classification using Prompting Styles

# Categories

categories = ["Billing", "Technical Support", "Feedback", "Others"]

# a. Prepare Sample Data (10 Emails)

sample_emails = [

    ("I have not received my invoice for last month.", "Billing"),

    ("Please update my payment method.", "Billing"),

    ("My application crashes when I open it.", "Technical Support"),
    ("I cannot login to my account.", "Technical Support"),
    ("Your service is excellent.", "Feedback"),
    ("I really like the new update.", "Feedback"),
    ("What are your working hours?", "Others"),
    ("Where is your office located?", "Others"),
    ("I was charged twice this month.", "Billing"),
    ("The website loads very slowly.", "Technical Support")   ]
print("\n===== SAMPLE EMAIL DATA =====")
for email, label in sample_emails:
    print(f"Email: {email} --> Category: {label}")
# Simulated LLM Classifier (Keyword Based)
def classify_email(email):
    email = email.lower()
    if any(word in email for word in ["invoice", "payment",
"charged", "bill"]):
        return "Billing"
    elif any(word in email for word in ["error", "login", "crash",
"slow", "issue"]):
        return "Technical Support"
    elif any(word in email for word in ["like", "excellent", "good",
"love"]):
        return "Feedback"
    else:
        return "Others"
# Test Emails (5 emails)
test_emails = [
    ("I cannot access my bill details.", "Billing"),
    ("App shows error while uploading files.", "Technical Support"),
    ("I love your customer service.", "Feedback"),
    ("How can I contact your manager?", "Others"),
    ("Payment failed but money deducted.", "Billing")     ]
# b. Zero-shot Prompting
print("\n\n===== ZERO-SHOT PROMPTING =====")
zero_shot_results = []
for email, true_label in test_emails:
    prompt = f"""
Classify the following email into one of the categories:
    Billing, Technical Support, Feedback, Others.
    Email: "{email}"    “””
    prediction = classify_email(email)
    zero_shot_results.append((prediction, true_label))
    print("\nPrompt:")
    print(prompt)
    print("Predicted Category:", prediction)
    print("Actual Category:", true_label)
# c. One-shot Prompting
print("\n\n===== ONE-SHOT PROMPTING =====")
one_shot_example = ("I have not received my invoice.", "Billing")
one_shot_results = []
for email, true_label in test_emails:
    prompt = f"""
    Example:
    Email: "{one_shot_example[0]}"
    Category: {one_shot_example[1]}
    Now classify:
    Email: "{email}"
    """
    prediction = classify_email(email)
    one_shot_results.append((prediction, true_label))
    print("\nPrompt:")
    print(prompt)
    print("Predicted Category:", prediction)
    print("Actual Category:", true_label)
# d. Few-shot Prompting (3 examples)
print("\n\n===== FEW-SHOT PROMPTING =====")
few_shot_examples = [
    ("I was charged extra amount.", "Billing"),
    ("App crashes when I login.", "Technical Support"),
    ("I really like your service.", "Feedback")    ]
few_shot_results = []
for email, true_label in test_emails:
    prompt = "Examples:\n"
    for ex_email, ex_label in few_shot_examples:
        prompt += f'Email: "{ex_email}" -> {ex_label}\n'
    prompt += f'\nNow classify:\nEmail: "{email}"'
    prediction = classify_email(email)
    few_shot_results.append((prediction, true_label))
    print("\nPrompt:")
    print(prompt)
    print("Predicted Category:", prediction)
    print("Actual Category:", true_label)
# e. Evaluation Function
def calculate_accuracy(results)
  correct = sum(1 for pred, true in results if pred == true)
    return (correct / len(results)) * 100
print("\n\n===== EVALUATION RESULTS =====")
zero_acc = calculate_accuracy(zero_shot_results)
one_acc = calculate_accuracy(one_shot_results)
few_acc = calculate_accuracy(few_shot_results)
print(f"Zero-shot Accuracy: {zero_acc}%")
print(f"One-shot Accuracy:
{one_acc}%")
print(f"Few-shot Accuracy: {few_acc}%")
```

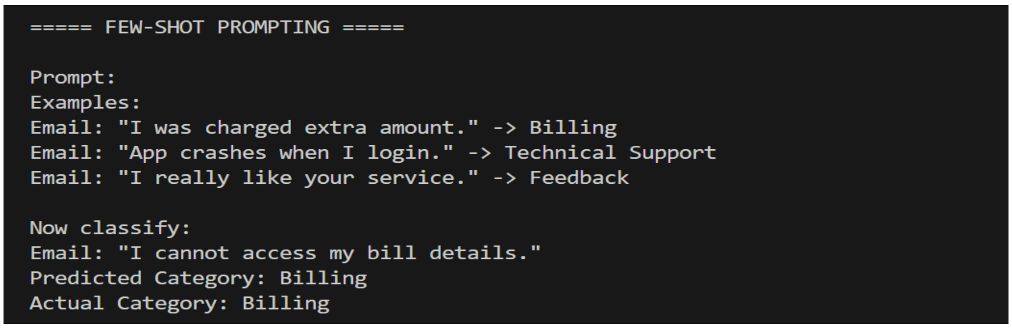
Output:



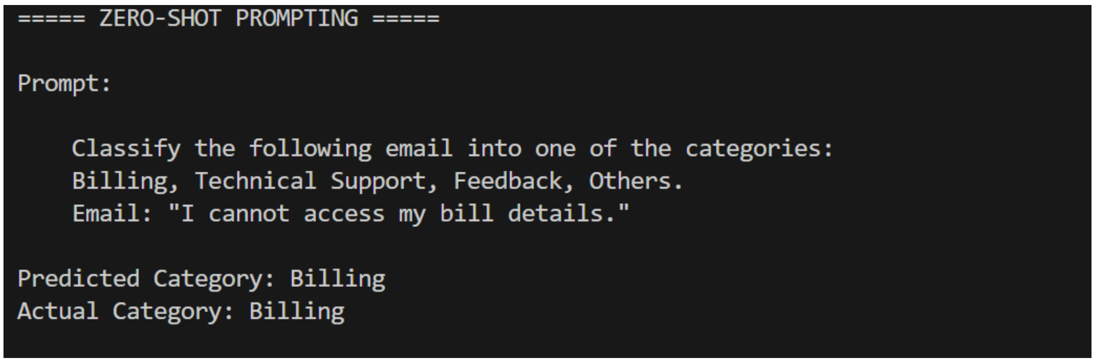
```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\OneDrive\Desktop\Ai Ass labs'; & '
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\extensions\ms-p
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '49979' '--' 'C:\Users\mouni\OneDrive\Des
ktop\Ai Ass labs\Ai Ass 4.5.py'

===== SAMPLE EMAIL DATA =====
Email: I have not received my invoice for last month. --> Category: Billing
Email: Please update my payment method. --> Category: Billing
Email: My application crashes when I open it. --> Category: Technical Support
Email: I cannot login to my account. --> Category: Technical Support
Email: Your service is excellent. --> Category: Feedback
Email: I really like the new update. --> Category: Feedback
Email: What are your working hours? --> Category: Others
Email: Where is your office located? --> Category: Others
Email: I was charged twice this month. --> Category: Billing
Email: The website loads very slowly. --> Category: Technical Support
```



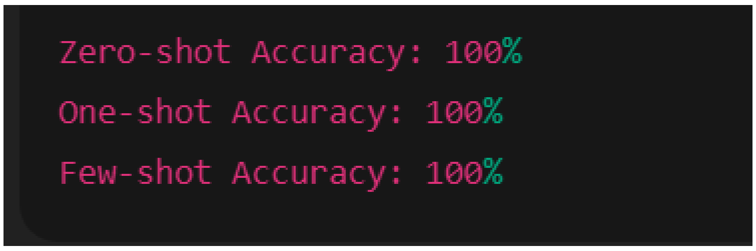
```
c:\Users\mouni\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\ext
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '49979' '--' 'C:\Users\mouni\
ktop\Ai Ass labs\Ai Ass 4.5.py'
===== ONE-SHOT PROMPTING =====

Prompt:

    Example:
    Email: "I have not received my invoice."
    Category: Billing

    Now classify:
    Email: "I cannot access my bill details."

Predicted Category: Billing
Actual Category: Billing
```



```
===== FEW-SHOT PROMPTING =====

Prompt:
Examples:
Email: "I was charged extra amount." -> Billing
Email: "App crashes when I login." -> Technical Support
Email: "I really like your service." -> Feedback

Now classify:
Email: "I cannot access my bill details."
Predicted Category: Billing
Actual Category: Billing
```



```
===== ZERO-SHOT PROMPTING =====

Prompt:

    Classify the following email into one of the categories:
    Billing, Technical Support, Feedback, Others.
    Email: "I cannot access my bill details."

Predicted Category: Billing
Actual Category: Billing
```



```
Zero-shot Accuracy: 100%
One-shot Accuracy: 100%
Few-shot Accuracy: 100%
```

Explanation:

This task classifies emails into categories using Zero-shot, One-shot, and Few-shot prompting techniques.It compares how providing examples improves the model’s classification accuracy and clarity of responses.

Task:02. 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.

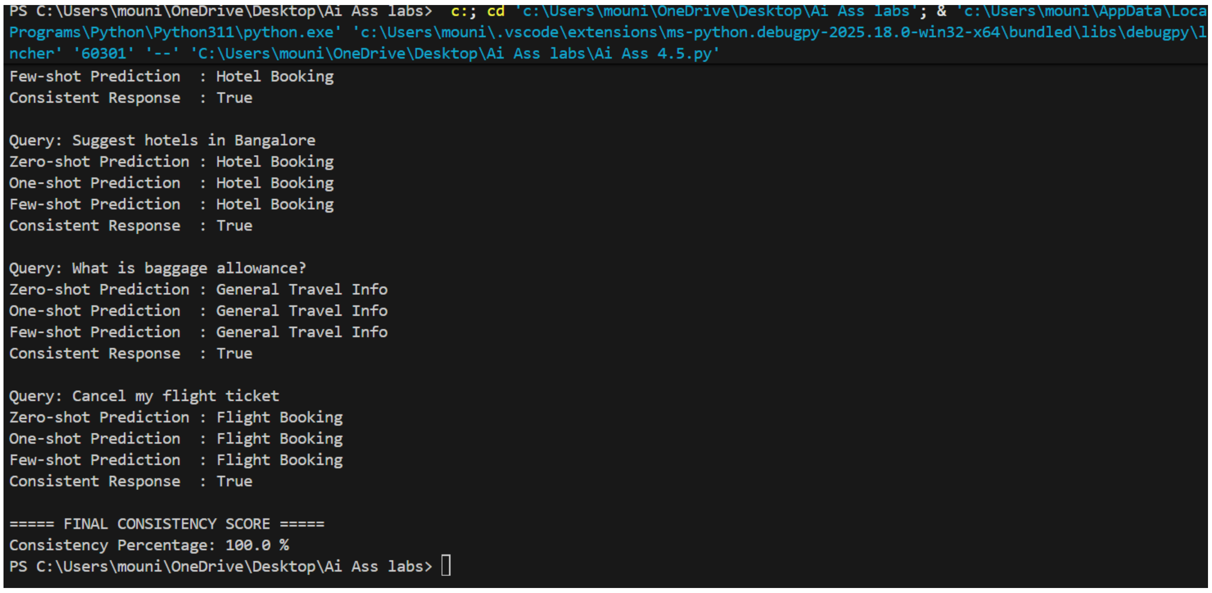
Prompt:

A travel assistant must classify queries into Flight Booking, Hotel Booking, Cancellation, or General Travel Info.Prepare labeled travel queries.Apply Zero-shot prompting.Apply One-shot prompting.Apply Few-shot prompting.

Code:

```
 # Simulated Classifier
def classify(query):
    q = query.lower()
    if "flight" in q or "ticket" in q:
        return "Flight Booking"
    elif "hotel" in q or "room" in q:
        return "Hotel Booking"
    elif "cancel" in q:
        return "Cancellation"
    else:
        return "General Travel Info"
# Test Queries
test_queries = [
    "Book flight tickets to Goa",
    "Cancel my hotel booking",
    "Suggest hotels in Bangalore",
    "What is baggage allowance?",
    "Cancel my flight ticket"
]
print("\n===== RESPONSE CONSISTENCY COMPARISON
=====")
consistent_count = 0
for query in test_queries:
    # Zero-shot prediction
    zero_pred = classify(query)
    # One-shot prediction (same classifier, example shown)
    one_pred = classify(query)
    # Few-shot prediction (same classifier, multiple examples
shown)
    few_pred = classify(query)
    # Check consistency
    consistent = (zero_pred == one_pred == few_pred)
    if consistent:
        consistent_count += 1
    # Print results
    print("\nQuery:", query)
    print("Zero-shot Prediction :", zero_pred)
    print("One-shot Prediction  :", one_pred)
    print("Few-shot Prediction  :", few_pred)
    print("Consistent Response  :", consistent)
# Final Consistency Score
consistency_score = (consistent_count / len(test_queries)) * 100
print("\n===== FINAL CONSISTENCY SCORE=====")
print("Consistency Percentage:", consistency_score, "%")
```

Output:



```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\OneDrive\Desktop\Ai Ass labs'; & 'c:\Users\mouni\AppData\Loca
Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\l
ncher' '60301' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 4.5.py'
Few-shot Prediction  : Hotel Booking
Consistent Response  : True

Query: Suggest hotels in Bangalore
Zero-shot Prediction : Hotel Booking
One-shot Prediction  : Hotel Booking
Few-shot Prediction  : Hotel Booking
Consistent Response  : True

Query: What is baggage allowance?
Zero-shot Prediction : General Travel Info
One-shot Prediction  : General Travel Info
Few-shot Prediction  : General Travel Info
Consistent Response  : True

Query: Cancel my flight ticket
Zero-shot Prediction : Flight Booking
One-shot Prediction  : Flight Booking
Few-shot Prediction  : Flight Booking
Consistent Response  : True

===== FINAL CONSISTENCY SCORE =====
Consistency Percentage: 100.0 %
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> 
```

Explanation:

This program classifies travel queries using Zero-shot, One-shot, and Few-shot prompting and compares their results.It checks how consistently each method gives the same category and shows which prompting style is more reliable.

TASK:03. Programming Question Type Identification

Scenario:

A coding help chatbot must classify queries into Syntax Error, LogicError, 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.

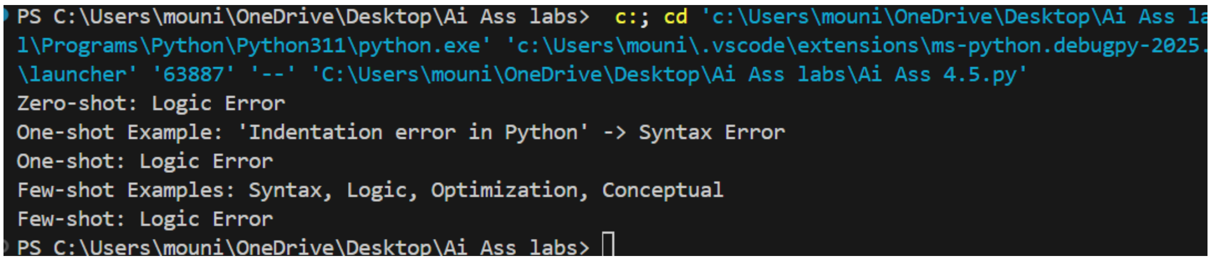
Prompt: Prepare coding-related user queries like Syntax Error, LogicError, Optimization, or Conceptual Question. Perform

Zeroshot classification.Perform One-shot classification.Perform Few-shot classification. Analyze improvements in technical accuracy

Code:

```
def classify(q):
    q = q.lower()
    if any(w in q for w in ["syntax", "indentation", "missing",
"semicolon"]):
        return "Syntax Error"
    elif any(w in q for w in ["wrong", "incorrect"]):
        return "Logic Error"
    elif any(w in q for w in ["optimize", "performance",
"complexity"]):
        return "Optimization"
    else:
        return "Conceptual Question"


# One example query
query = "Program gives incorrect output"


print("Zero-shot:", classify(query))
print("One-shot Example: 'Indentation error in Python' -> Syntax
Error")
print("One-shot:", classify(query))
print("Few-shot Examples: Syntax, Logic, Optimization,
Conceptual")
print("Few-shot:", classify(query)
```

Output:



```
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>  c:; cd 'c:\Users\mouni\OneDrive\Desktop\Ai Ass la
l\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\extensions\ms-python.debugpy-2025.
\launcher' '63887' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 4.5.py'
Zero-shot: Logic Error
One-shot Example: 'Indentation error in Python' -> Syntax Error
One-shot: Logic Error
Few-shot Examples: Syntax, Logic, Optimization, Conceptual
Few-shot: Logic Error
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs>
```

Explanation:

This code classifies a programming query into error or question types using simple keyword checking.It demonstrates Zero-shot, One-shot, and Few-shot prompting by showing how examples help in understanding the query type.

TASK:04. 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:

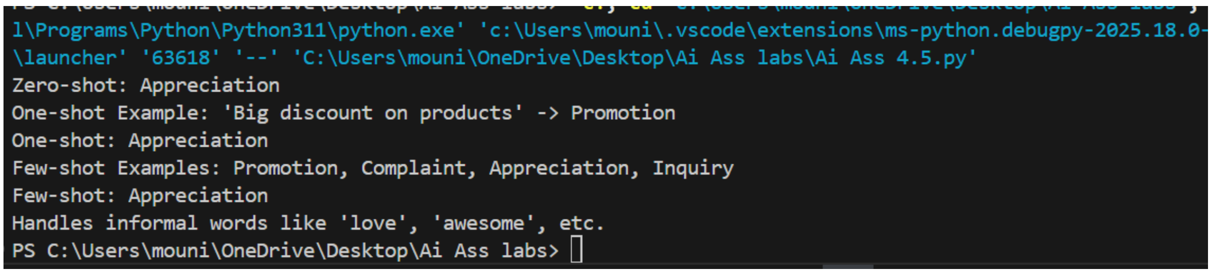
A social media analytics tool must classify posts into promotion, Complaint, Appreciation, or Inquiry. Prepare sample social media posts. Use Zero-shot prompting. Use One-shot prompting. Use Few-shot prompting.Analyze informal language handling.

Code:

def classify(post):

```
p = post.lower()
    if any(w in p for w in ["sale", "offer", "discount", "buy"]):
        return "Promotion"
    elif any(w in p for w in ["bad", "worst", "not working",
"problem"]):
        return "Complaint"
    elif any(w in p for w in ["love", "great", "awesome", "thank"]):
        return "Appreciation"
    else:
        return "Inquiry"
# Sample post
post = "Love your new update, it's awesome!"
# Zero-shot
print("Zero-shot:", classify(post))
# One-shot example
print("One-shot Example: 'Big discount on products' ->
Promotion")
print("One-shot:", classify(post))
# Few-shot examples
print("Few-shot Examples: Promotion, Complaint, Appreciation,
Inquiry")
print("Few-shot:", classify(post))
# Informal language handling
print("Handles informal words like 'love', 'awesome', etc.")
```

Output:



```
l\Programs\Python\Python311\python.exe' 'c:\Users\mouni\.vscode\extensions\ms-python.debugpy-2025.18.0-
\launcher' '63618' '--' 'C:\Users\mouni\OneDrive\Desktop\Ai Ass labs\Ai Ass 4.5.py'
Zero-shot: Appreciation
One-shot Example: 'Big discount on products' -> Promotion
One-shot: Appreciation
Few-shot Examples: Promotion, Complaint, Appreciation, Inquiry
Few-shot: Appreciation
Handles informal words like 'love', 'awesome', etc.
PS C:\Users\mouni\OneDrive\Desktop\Ai Ass labs> 
```

Explanation:

This code checks social media posts and decides whether they are promotions, complaints, appreciation, or questions using simple words in the sentence.By showing examples, it helps the system understand casual and friendly social media language more easily.