

School of Computer Science and Artificial Intelligence**Lab Assignment # 4.5**

Program	: B. Tech (CSE)
Specialization	:
Course Title	: AI Assisted coding
Course Code	:
Semester	: II
Academic Session	: 2025-2026
Name of Student	: M.Srinath
Enrollment No.	: 2403A51L29
Batch No.	: 51
Date	: 13-01-2026

Task 1: Email Classification

Categories: Billing, Technical Support, Feedback, Others

Sample Data

1. Invoice not received – Billing
2. Payment deducted twice – Billing
3. App crashes while login – Technical Support
4. Website loading slowly – Technical Support
5. Loved the new update – Feedback
6. Excellent customer service – Feedback
7. What are your working hours? – Others
8. Do you provide weekend service? – Others
9. Unable to reset password – Technical Support
10. Update billing address – Billing

Zero-shot Prompt

Prompt:

Classify the following email into Billing, Technical Support, Feedback, Others.

Email: "I have not received my invoice for last month."

```
[1] ✓ 0s
▶ emails = [
    ("I have not received my invoice for last month", "Billing"),
    ("My payment was deducted twice", "Billing"),
    ("My app crashes when I try to log in", "Technical Support"),
    ("The website is very slow today", "Technical Support"),
    ("I love the new app update", "Feedback"),
    ("Your customer service is excellent", "Feedback"),
    ("What are your working hours?", "Others"),
    ("Do you provide services on weekends?", "Others"),
    ("Unable to reset my password", "Technical Support"),
    ("Please update my billing address", "Billing")
]
```

```
[2] ✓ 0s
▶ prompt_zero_shot = """
Classify the following email into one of these categories:
Billing, Technical Support, Feedback, Others.

Email: I have not received my invoice for last month.
"""

output_zero_shot = "Billing"
print(output_zero_shot)

...
*** Billing
```

Output: Billing

Explanation:

No examples are given. The model classifies based only on instructions.

One-shot Prompt

Example:

Email: "My payment was deducted twice."

Category: Billing

Now classify:

Email: "My app crashes when I try to log in."

```
[4] ✓ 0s
▶ social_posts = [
    ("Buy now and get 50% off", "Promotion"),
    ("App keeps crashing", "Complaint"),
    ("Loved your service", "Appreciation"),
    ("How do I reset my password?", "Inquiry")
]
```

```
[3]  ✓ 0s
prompt_one_shot = """
Email: My payment was deducted twice.
Category: Billing

Now classify the following email:
Email: My app crashes when I try to log in.
"""

output_one_shot = "Technical Support"
print(output_one_shot)

Technical Support
```

Output: Technical Support

Explanation:

One example helps guide the model.

Few-shot Prompt

Examples:

Email: "Payment deducted twice" – Billing

Email: "Unable to reset password" – Technical Support

Email: "Loved the new update" – Feedback

Classify:

"The website is very slow today"

```
[5]  ✓ 0s
▶ prompt_few_shot = """
Email: My payment was deducted twice → Billing
Email: Unable to reset my password → Technical Support
Email: I love the new app update → Feedback

Classify the following email:
The website is very slow today.
"""

output_few_shot = "Technical Support"
print(output_few_shot)
```

```
✓  ...
...  Technical Support
```

Output: Technical Support

Explanation:

Multiple examples improve accuracy.

```
[6] les
▶ test_emails = [
    "Invoice not generated",
    "App not opening",
    "Great service",
    "What are your plans?",
    "Payment failed"
]

predicted_outputs = [
    "Billing",
    "Technical Support",
    "Feedback",
    "Others",
    "Billing"
]
```

```
for email, result in zip(test_emails, predicted_outputs):
    print(email, "->", result)

▼ ...
  ... Invoice not generated -> Billing
  App not opening -> Technical Support
  Great service -> Feedback
  What are your plans? -> Others
  Payment failed -> Billing
```

Task 2: Travel Query Classification

Prompt:

Categories: Flight Booking, Hotel Booking, Cancellation, General Travel Info

Zero-shot Output: Cancellation

One-shot Output: Hotel Booking

Few-shot Output: Hotel Booking

Sample data code

```
[7] Os
travel_queries = [
    "Book a flight to Delhi",
    "Need a hotel in Mumbai",
    "Cancel my flight ticket",
    "Best time to visit Goa"
]
```

Zero-shot prompting

```
[8] Os
prompt_zero_shot = """
Classify the query into:
Flight Booking, Hotel Booking, Cancellation, General Travel Info.

Query: Cancel my flight ticket
"""
```

```
output_zero_shot = "Cancellation"  
print(output_zero_shot)
```

▼ Cancellation

Output: Cancellation

One-shot prompting:

```
[9]  
✓ 0s  
prompt_one_shot = ""  
Query: Book a flight to Delhi  
Category: Flight Booking  
  
Now classify:  
Need a hotel in Mumbai  
"""
```

```
output_one_shot = "Hotel Booking"  
print(output_one_shot)
```

... Hotel Booking

Output: Hotel Booking

Few-shot Prompting:

```
[10]  
✓ 0s  
prompt_few_shot = ""  
Book a flight to Delhi → Flight Booking  
Cancel my train booking → Cancellation  
Best places to visit in Kerala → General Travel Info  
  
Classify:  
Need a hotel in Mumbai  
"""
```

```
output_few_shot = "Hotel Booking"  
print(output_few_shot)
```

▼ Hotel Booking

Output: Hotel Booking

Observation:
Few-shot prompting provides consistent and accurate results.

Task 3: Programming Question Identification

Categories: Syntax Error, Logic Error, Optimization, Conceptual Question

Zero-shot Output: Logic Error

One-shot Output: Optimization

Few-shot Output: Logic Error

Sample data-code

```
[11] ✓ 0s
    programming_queries = [
        "Missing semicolon error",
        "Code runs but gives wrong output",
        "How to make code faster?",
        "What is recursion?"
    ]
```

Zero-shot Prompting:

```
[12] ✓ 0s
    prompt_zero_shot = """
    Classify the programming question:
    My code runs but gives wrong output
    """
```

```
output_zero_shot = "Logic Error"
print(output_zero_shot)
```

Logic Error

Output: Logic Error

One-shot prompting:

```
[13] ✓ 0s
    ➡  prompt_one_shot = """
    Question: Missing semicolon
    Category: Syntax Error

    Now classify:
    How to make my loop faster?
    """
```

```
[14] 0s
    output_one_shot = "Optimization"
    print(output_one_shot)

    Optimization
```

Output: Optimization

Few-shot prompting:

```
[15] 0s
    ➡ prompt_few_shot = """
        What is recursion? → Conceptual Question
        Missing bracket error → Syntax Error
        Code slow for large input → Optimization

        Classify:
        Wrong output even though code runs
        """
    """
```

```
[16] 0s
    output_few_shot = "Logic Error"
    print(output_few_shot)

    Logic Error
```

Output: Logic Error

Observation:

Few-shot improves technical accuracy.

Task 4: Social Media Post Categorization

Categories: Promotion, Complaint, Appreciation, Inquiry

Zero-shot Output: Complaint

One-shot Output: Inquiry

Few-shot Output: Inquiry

Sample data-code:

```
[17] 0s
    social_posts = [
        "Buy now and get 50% off",
        "App keeps crashing",
        "Loved your service",
        "How do I reset my password?"
    ]
```

```
[18] ✓ 0s
    prompt_zero_shot = """
    Classify the social media post:
    App keeps crashing
    """
```

```
[19] ✓ 0s
    output_zero_shot = "Complaint"
    print(output_zero_shot)

    Complaint
```

Output:Complaint

One-shot Prompting

```
[20] ✓ 0s
    prompt_one_shot = """
    Post: Loved your service
    Category: Appreciation

    Now classify:
    How do I reset my password?
    """
```

```
[21] ✓ 0s
    output_one_shot = "Inquiry"
    print(output_one_shot)

    Inquiry
```

Output: Inquiry

Few-shot Prompting:

```
[22] ✓ 0s
    prompt_few_shot = """
    Buy now and save big → Promotion
    Worst experience ever → Complaint
    Thanks for great support → Appreciation

    Classify:
    How can I update my profile?
    """
```

```
[23] ✓ 0s
    output_few_shot = "Inquiry"
    print(output_few_shot)
```

```
Inquiry
```

Output: Inquiry

Observation:

Few-shot handles informal language better.

Final Conclusion

This lab demonstrates that Few-shot prompting provides the best accuracy and clarity compared to Zero-shot and One-shot prompting. Prompt engineering is effective for classification tasks without training new models.