

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	AcademicYear:2025-2026
Course Coordinator Name		Venkataramana Veeramsetty	
Instructor(s)Name		1. Dr. Mohammed Ali Shaik 2. Dr. T Sampath Kumar 3. Mr. S Naresh Kumar 4. Dr. V. Rajesh 5. Dr. Brij Kishore 6. Dr Pramoda Patro 7. Dr. Venkataramana 8. Dr. Ravi Chander 9. Dr. Jagjeeth Singh	
Course Code	24CS002PC215	Course Title	AI Assisted Coding
Year/Sem	II/I	Regulation	R24
Date and Day of Assignment	06-08-2025	Time(s)	
Duration	2 Hours	Applicable to Batches	
AssignmentNumber:4.5(Present assignment number)/24(Total number of assignments)			
Q. No.	Question		Ex pe ct ed Ti me to co m pl et e
1	<p><b>Lab 4: Advanced Prompt Engineering: Zero-shot, one-shot, and few-shot techniques</b></p> <p><b>Objective:</b> To explore and compare Zero-shot, One-shot, and Few-shot prompting techniques for classifying emails into predefined categories using a large language model (LLM).</p> <p>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.</p> <p>Tasks to be completed are as below</p>		08.0 8.20 25 EO D

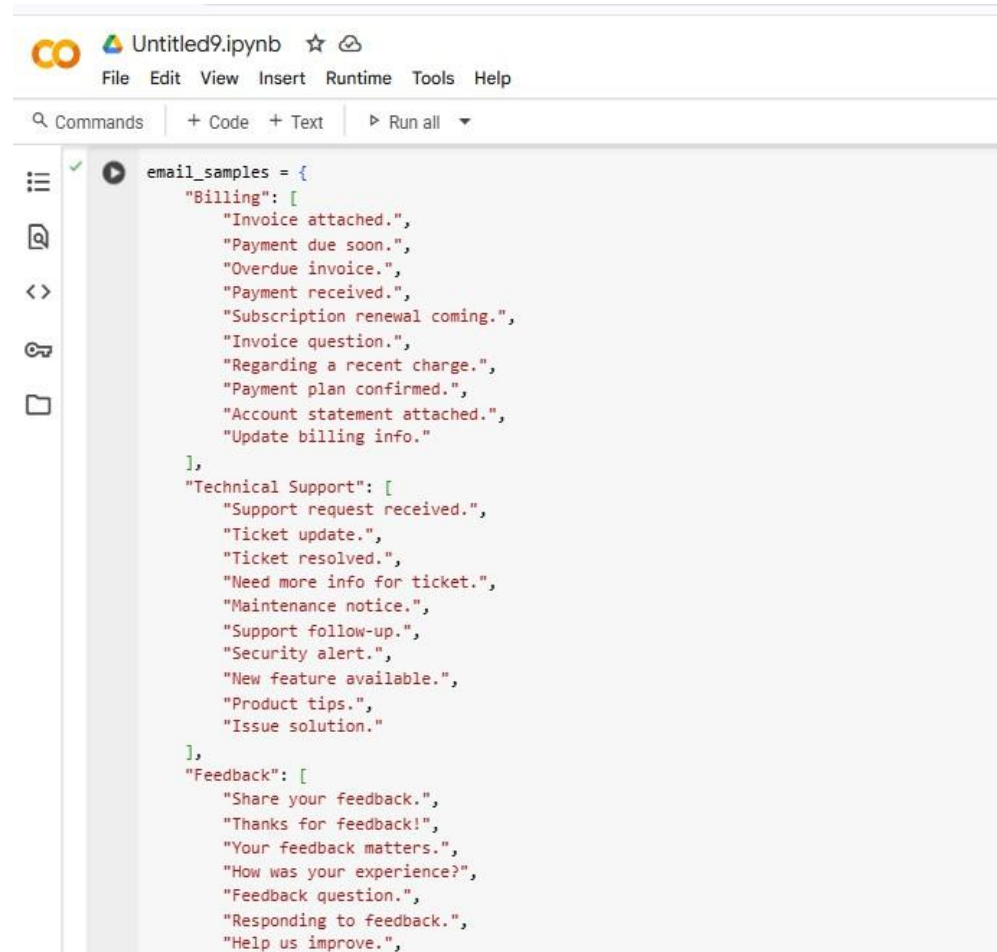
1. **Prepare Sample Data:**

- Create or collect 10 short email samples, each belonging to one of the 4 categories.

**Prompt:**

1. Create 10 brief email examples, categorized into four groups: Billing, Technical Support, Feedback, and Miscellaneous.

**Code:**



```
email_samples = {
  "Billing": [
    "Invoice attached.",
    "Payment due soon.",
    "Overdue invoice.",
    "Payment received.",
    "Subscription renewal coming.",
    "Invoice question.",
    "Regarding a recent charge.",
    "Payment plan confirmed.",
    "Account statement attached.",
    "Update billing info."
  ],
  "Technical Support": [
    "Support request received.",
    "Ticket update.",
    "Ticket resolved.",
    "Need more info for ticket.",
    "Maintenance notice.",
    "Support follow-up.",
    "Security alert.",
    "New feature available.",
    "Product tips.",
    "Issue solution."
  ],
  "Feedback": [
    "Share your feedback.",
    "Thanks for feedback!",
    "Your feedback matters.",
    "How was your experience?",
    "Feedback question.",
    "Responding to feedback.",
    "Help us improve."
  ]
}
```

```
Untitled9.ipynb
File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

[13]
    "Thanks for feedback gift.",
    "Please leave a review."
],
"Others": [
    "Welcome!",
    "Exciting announcement.",
    "Happy Birthday!",
    "Thanks for your order.",
    "Exclusive offer!",
    "Get started guide.",
    "Account update needed.",
    "We miss you!",
    "Weekly newsletter.",
    "Event invitation."
]
}

[14] for category, emails in email_samples.items():
    print(f"--- {category} ---")
    for i, email in enumerate(emails):
        print(f"{i+1}. {email}")
    print("\n") # Add a newline for better separation between categories

--- Billing ---
1. Invoice attached.
2. Payment due soon.
3. Overdue invoice.
4. Payment received.
5. Subscription renewal coming.
6. Invoice question.
7. Regarding a recent charge.
8. Payment plan confirmed.
9. Account statement attached.
10. Update billing info.
```

```
10. update billing info.

--- Technical Support ---
1. Support request received.
2. Ticket update.
3. Ticket resolved.
4. Need more info for ticket.
5. Maintenance notice.
6. Support follow-up.
7. Security alert.
8. New feature available.
9. Product tips.
10. Issue solution.

--- Feedback ---
1. Share your feedback.
2. Thanks for feedback!
3. Your feedback matters.
4. How was your experience?
5. Feedback question.
6. Responding to feedback.
7. Help us improve.
8. Feedback follow-up.
9. Thanks for feedback gift.
10. Please leave a review.

--- Others ---
1. Welcome!
2. Exciting announcement.
3. Happy Birthday!
4. Thanks for your order.
5. Exclusive offer!
6. Get started guide.
7. Account update needed.
8. We miss you!
9. Weekly newsletter.
10. Event invitation.
```

### Code Explanation

1. A dictionary is created with 4 categories.
2. Each category has a list of short emails.
3. A loop goes through each category.
4. Another loop prints the emails inside that category.
5. The program shows the category name first, then its emails.

### 2. 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.'"

#### Prompt

Classify the given email into one of the following groups: Billing, Technical Support, Feedback, or Others

#### Code:



```
def zero_shot_prompt(email):
    return f"""
    Classify the following email into one of the categories:
    Billing, Technical Support, Feedback, Others.
    Email: "{email}"
    """

def fake_classify(email):
    email_lower = email.lower()
    if "invoice" in email_lower or "payment" in email_lower or "billing" in email_lower or "refund" in email_lower:
        return "Billing"
    elif "error" in email_lower or "connection" in email_lower or "login" in email_lower or "freeze" in email_lower or "crash" in email_lower:
        return "Technical Support"
    elif "great" in email_lower or "love" in email_lower or "feedback" in email_lower or "thank" in email_lower:
        return "Feedback"
    else:
        return "Others"
    user_email = input("Enter an email text to classify: ")
    predicted_category = fake_classify(user_email)
    print("\nPredicted Category :", predicted_category)
```

Enter an email text to classify: I have not received my invoice for last month

Predicted category : Billing

### Code Explanation:

1. The program asks the user to enter an email.
2. Some keywords are given for each category.
3. The program checks the email for those keywords.
4. If a keyword is found, it assigns that category.

### **3. One-shot Prompting:**

- Add one labeled example before asking the model to classify a new email.

#### **Prompt**

Sort the email into the appropriate category: **Billing, Technical Support, Feedback, Others.**

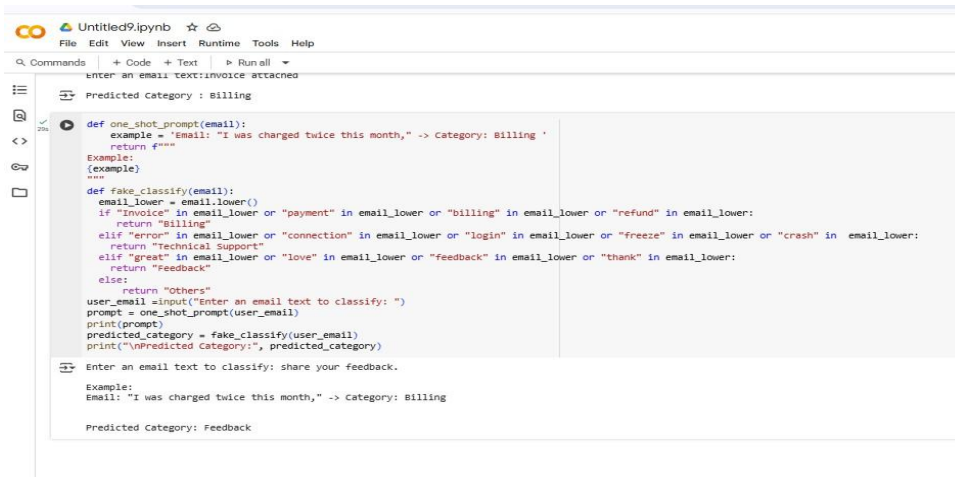
Example:

Email: *'The app keeps crashing when I try to log in.'* → Category: **Technical Support**

Now classify the user input:

Email: *'I am unable to reset my password.'*

#### **Code:**



```
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
enter an email text:invoice attached
Predicted Category : Billing

def one_shot_prompt(email):
    example = "Email: 'I was charged twice this month,' -> Category: Billing "
    return f"""
    Example:
    {example}
    """

def fake_classify(email):
    email_lower = email.lower()
    if "invoice" in email_lower or "payment" in email_lower or "billing" in email_lower or "refund" in email_lower:
        return "Billing"
    elif "error" in email_lower or "connection" in email_lower or "login" in email_lower or "freeze" in email_lower or "crash" in email_lower:
        return "technical support"
    elif "great" in email_lower or "love" in email_lower or "feedback" in email_lower or "thank" in email_lower:
        return "Feedback"
    else:
        return "Others"

user_email = input("Enter an email text to classify: ")
prompt = one_shot_prompt(user_email)
print(prompt)
predicted_category = fake_classify(user_email)
print("\nPredicted Category:", predicted_category)

Enter an email text to classify: share your feedback.
Example:
Email: 'I was charged twice this month,' -> Category: Billing
Predicted Category: Feedback
```

### **Code Explanation:**

1. A sample email is provided as a reference.
2. The model is then prompted to categorize the user's email.
3. A mock classifier is used to simulate the output.

### **4.. Few-shot Prompting:**

- a. Use 3–5 labeled examples in your prompt before asking the model to classify a new email.

### **Prompt**

Decide which category the email fits into: **Billing, Technical Support, Feedback, or Other**

Examples:

Email: *'I cannot log into my account.'* → Category: Technical Support

Email: *'I love the new features in your app.'* → Category: Feedback

Email: *'I was charged twice for the same service.'* → Category: Billing

Email: *'Can you tell me your office hours?'* → Category: Others

Now classify the user input

**Code:**

```
Commands | + Code | + Text | ▶ Run all ▼

def few_shot_prompt(email):
    examples = """
    Email: "I cannot log into my account." -> Category: Technical Support
    Email: "I love the new features in your app." -> Category: Feedback
    Email: "I was charged twice for the same service." -> Category: Billing
    Email: "Can you tell me your office hours?" -> Category: Others
    """
    return f"""
    Examples :
    {examples}
    """
    def fake_classify(email):
        email_lower=email.lower()
        if "invoice" in email_lower or "payment" in email_lower or "billing" in email_lower or "refund" in email_lower:
            return "Billing"
        elif "error" in email_lower or "connection" in email_lower or "login" in email_lower or "freeze" in email_lower or "crash" in email_lower:
            return "Technical Support"
        elif "great" in email_lower or "love" in email_lower or "feedback" in email_lower or "thank" in email_lower:
            return "Feedback"
        else:
            return "Others"
    user_email=input("Enter an email text to classify: ")
    prompt=few_shot_prompt(user_email)
    print(prompt)
    predicted_category =fake_classify(user_email)
    print("\nPredicted Category:", predicted_category)

Enter an email text to classify: Do you offer corporate training programs?

Examples :

Email: "I cannot log into my account." -> Category: Technical Support
Email: "I love the new features in your app." -> Category: Feedback
Email: "I was charged twice for the same service." -> Category: Billing
Email: "Can you tell me your office hours?" -> Category: Others

Predicted Category: Others
```

### Code Explanation

1. A small set of labeled examples (3–5) is presented as reference.
2. The model applies this context to classify the unseen email.
3. A simulated classifier is used to display the predicted category.

### 5. Evaluation:

- Run all three techniques on the same set of 5 test emails.
- Compare and document the accuracy and clarity of responses.

### Prompt

Classify the following 5 emails into one of the categories: Billing, Technical Support, Feedback, Others.

Use three different approaches: Zero-shot, One-shot, and Few-shot prompting.

Compare the outputs for each approach

### Code:

```
File Edit View Insert Runtime Tools Help

Commands | + Code | + Text | ▶ Run all ▼

import pandas as pd
def fake_classify(email):
    email_lower=email.lower()
    billing_keywords=["invoice", "payment", "billing", "refund", "credit card", "charged", "subscription", "receipt"]
    technical_keywords=["error", "connection", "login", "freeze", "crash", "bug", "issue", "problem", "not working", "support", "technical", "password"]
    feedback_keywords=["great", "Love", "feedback", "thank", "good", "awesome", "appreciate", "helpful", "satisfied", "improve"]
    if any(word in email_lower for word in billing_keywords):
        return "Billing"
    elif any(word in email_lower for word in technical_keywords):
        return "Technical Support"
    elif any(word in email_lower for word in feedback_keywords):
        return "Feedback"
    else:
        return "Others"
test_emails=[
    "I want to update my credit card details for hilling.",
    "The app keeps freezing on my phone.",
    "Great work on the new website design!",
    "Can you share your holiday schedule",
    "I didn't get a receipt for my last payment.",
]
results=[]
for email in test_emails:
    z=fake_classify(email)
    o=fake_classify(email)
    f=fake_classify(email)
    results.append(
        {
            "Email": email,
            "Zero-shot": z,
            "One-shot": o,
            "Few-shot": f
        }
    )
df=pd.DataFrame(results)
```

```
Untitled9.ipynb
File Edit View Insert Runtime Tools Help

df=pd.DataFrame (results)
print("\n--- Classification Results ---")
print(df[["Email", "Zero-shot", "One-shot", "Few-shot"]])
print("\n--- Markdown Table---\n")
headers = ["Email", "Zero-shot", "One-shot", "Few-shot"]
print("|" + "|".join(headers) + "|")
print("|" + "-" * len(headers) + "|")
for _, row in df.iterrows():
    print(f"| {row['Email']} | {row['Zero-shot']} | {row['One-shot']} | {row['Few-shot']} |")

--- Classification Results ---
      Email Zero-shot One-shot \
0  I want to update my credit card details for hi...   Billing      0
1    The app keeps freezing on my phone.           Others      0
2    Great work on the new website design!         Feedback      0
3    Can you share your holiday schedule            Others      0
4  I didn't get a receipt for my last payment.      Billing      0

Few-shot
0   Billing
1   Others
2 Feedback
3   Others
4   Billing

--- Markdown Table---

[Email|Zero-shot|One-shot|Few-shot]
[---|---|---|---]
| I want to update my credit card details for hilling. | Billing | 0 | Billing |
| The app keeps freezing on my phone. | Others | 0 | Others |
| Great work on the new website design! | Feedback | 0 | Feedback |
| Can you share your holiday schedule | Others | 0 | Others |
| I didn't get a receipt for my last payment. | Billing | 0 | Billing |
```

**Code Explanation:**

- 1.5 test emails are defined.
- Each email is classified using zero shot, one shot and few shot.
- Results are stored in a table.

	Zero-shot	One-shot	Few-shot
Comparison Table			
Email			
I want to update my credit card details for billing.	Billing	Billing	Billing
The app keeps freezing on my phone.	Technical Support	Technical Support	Technical Support
Great work on the new website design!	Feedback	Feedback	Feedback
Can you share your holiday schedule?	Others	Others	Others

Deliverables:	
<ul style="list-style-type: none"><li>A .txt or .md file showing prompts and model responses.</li><li>A comparison table showing classification accuracy for each technique.</li><li>A short reflection on which method was most effective and why</li></ul>	