

2303a51602

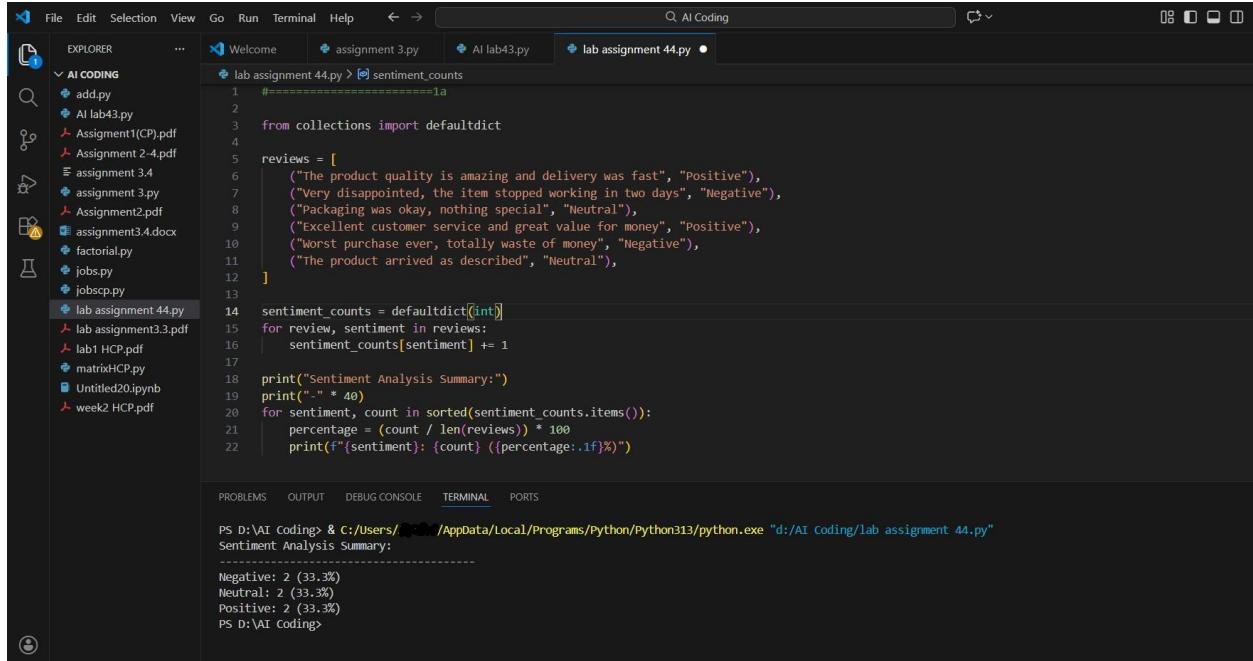
## Lab assignment-4.4

### 1. Sentiment Classification for Customer Reviews Scenario:

An e-commerce platform wants to analyze customer reviews and classify them into Positive, Negative, or Neutral sentiments using prompt engineering

Tasks:

#### a) Prepare 6 short customer reviews mapped to sentiment labels.



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

- File Explorer (Left):** Shows files in the "AI CODING" folder, including "add.py", "AI lab43.py", "Assignment1(CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "assignment34.docx", "factorial.py", "jobs.py", "jobsdp.py", "lab assignment 44.py" (selected), "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.ipynb", and "week2 HCP.pdf".
- Code Editor (Center):** Displays Python code for sentiment analysis. The code reads reviews from a list and counts the occurrences of Positive, Negative, and Neutral sentiments using a defaultdict. It then prints a summary with the count and percentage for each sentiment.

```
1 #=====
2
3 from collections import defaultdict
4
5 reviews = [
6     ("The product quality is amazing and delivery was fast", "Positive"),
7     ("Very disappointed, the item stopped working in two days", "Negative"),
8     ("Packaging was okay, nothing special", "Neutral"),
9     ("Excellent customer service and great value for money", "Positive"),
10    ("Worst purchase ever, totally waste of money", "Negative"),
11    ("The product arrived as described", "Neutral"),
12 ]
13
14 sentiment_counts = defaultdict(int)
15 for review, sentiment in reviews:
16     sentiment_counts[sentiment] += 1
17
18 print("Sentiment Analysis Summary:")
19 print("." * 40)
20 for sentiment, count in sorted(sentiment_counts.items()):
21     percentage = (count / len(reviews)) * 100
22     print(f"{sentiment}: {count} ({percentage:.1f}%)")
```

- Terminal (Bottom):** Shows the command run in the terminal: "PS D:\AI Coding> & C:/Users/.../AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py" Sentiment Analysis Summary: ----- Negative: 2 (33.3%) Neutral: 2 (33.3%) Positive: 2 (33.3%) PS D:\AI Coding>"

#### b) Design a Zero-shot prompt to classify sentiment.

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

- File Explorer:** Shows files in the "AI CODING" folder, including "add.py", "AI lab43.py", "Assignment1(CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobsccp.py", "lab assignment 44.py", "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.ipynb", and "week2 HCP.pdf".
- Code Editor:** The active file is "lab assignment 44.py". The code performs sentiment analysis on a review string. It defines positive and negative word lists, converts the review to lowercase, counts words in each list, and determines the sentiment based on the counts.
- Terminal:** The terminal shows the command "PS D:\AI Coding & C:/Users/... /AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"" being run, followed by the output "Sentiment Analysis Summary:" and the results: Negative: 2 (33.3%), Neutral: 2 (33.3%), Positive: 2 (33.3%).
- Status Bar:** Shows the current file is "lab assignment 44.py", and other tabs like "assignment 3.py" and "AI lab43.py" are open.

### 1c) Design a One-shot prompt with one labeled example.

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

- File Explorer:** Same as the previous screenshot, showing files in the "AI CODING" folder.
- Code Editor:** The active file is "lab assignment 44.py". The code has been modified to analyze a single review: "Excellent customer service and great value for money".
- Terminal:** The terminal shows the command "PS D:\AI Coding & C:/Users/... /AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"" being run, followed by the output "Sentiment Analysis Summary:" and the results: Negative: 2 (33.3%), Neutral: 2 (33.3%), Positive: 2 (33.3%).
- Status Bar:** Shows the current file is "lab assignment 44.py", and other tabs like "assignment 3.py" and "AI lab43.py" are open.

**1d) Design a Few-shot prompt with 3–5 labeled examples.**

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: AI CODING, add.py, AI lab43.py, Assignment1(CP).pdf, Assignment 2-4.pdf, assignment 3.4, assignment 3.py, Assignment2.pdf, assignment3.4.docx, factorial.py, jobs.py, jobspp.py, lab assignment 44.py, lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, week2 HCP.pdf.
- CODE** tab: A Python script named "lab assignment 44.py". The code defines a function "classify\_sentiment" that takes a review as input and returns "Positive", "Negative", or "Neutral" based on word counts. It includes a test case for a review about product quality.
- TERMINAL** tab: Shows command-line output from running the script with different reviews. The output indicates sentiment counts and percentages: Neutral: 2 (33.3%), Positive: 2 (33.3%).
- OUTPUT** tab: Shows logs for the Python environment, including module imports like powershell, Python, and Python.

**1e) Compare the outputs and discuss accuracy differences.**

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: AI CODING, add.py, AI lab43.py, Assignment1(CP).pdf, Assignment 2-4.pdf, assignment 3.4, assignment 3.py, Assignment2.pdf, assignment3.4.docx, factorial.py, jobs.py, jobspp.py, lab assignment ... (highlighted), lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, week2 HCP.pdf.
- CODE** tab: A Python script named "lab assignment 44.py". The code defines a variable "data" containing a dictionary with keys "Technique", "Accuracy", and "Reason".
- TERMINAL** tab: Shows command-line output from running the script. The output indicates sentiment: Positive for the first review and Negative for the second.
- OUTPUT** tab: Shows logs for the Python environment, including module imports like powershell, Python, and Python.

**2. Email Priority Classification**

**Scenario:**

A company wants to automatically prioritize incoming emails into High Priority, Medium Priority, or Low Priority

**2a) Create 6 sample email messages with priority labels.**

The screenshot shows the Visual Studio Code (VS Code) interface. The left sidebar has a tree view labeled "EXPLORER" with sections for "AI CODING" and "JOBS". The "AI CODING" section contains files like "add.py", "AI lab43.py", "Assignment1(CP).pdf", etc. The "JOBS" section contains "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobscp.py", and "lab assignment 44.py". The main area shows a code editor with the following Python script:

```
9  #=====
10 emails = [
11     {"email": "Server is down and business operations stopped", "priority": "High"},
12     {"email": "Client meeting scheduled for tomorrow", "priority": "High"},  
13     {"email": "Please review the attached report when free", "priority": "Medium"},  
14     {"email": "Need update on project status", "priority": "Medium"},  
15     {"email": "Team lunch invitation", "priority": "Low"},  
16     {"email": "Newsletter subscription confirmation", "priority": "Low"},  
17 ]  
18  
19 for email in emails:  
20     print(f"Email: {email['email']} | Priority: {email['priority']}")
```

Below the code editor, there are tabs for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS". The "TERMINAL" tab is selected, showing the output of the Python script:

```
Traceback (most recent call last):  
  File "d:\AI Coding\lab assignment 44.py", line 1, in <module>  
    import pandas as pd  
ModuleNotFoundError: No module named 'pandas'  
PS D:\AI Coding> & c:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
Email: Server is down and business operations stopped | Priority: High  
Email: Client meeting scheduled for tomorrow | Priority: High  
Email: Please review the attached report when free | Priority: Medium  
Email: Need update on project status | Priority: Medium  
Email: Team lunch invitation | Priority: Low  
Email: Newsletter subscription confirmation | Priority: Low  
PS D:\AI Coding>
```

## 2b) Perform intent classification using Zero-shot prompting

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "AI CODING" containing files like "add.py", "AI lab43.py", "Assignment1(CP).pdf", etc.
- Code Editor:** Displays a Python file "lab assignment 44.py" with the following code:

```
7 #  
8 #  
9 #  
10 priority = "High"
```

A tooltip above the code editor says "Generate code" and "Add Context...".
- Terminal:** Shows the command line output:

```
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>  
    import pandas as pd  
ModuleNotFoundError: No module named 'pandas'  
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
Email: Server is down and business operations stopped | Priority: High  
Email: Client meeting scheduled for tomorrow | Priority: High  
Email: Please review the attached report when free | Priority: Medium  
Email: Need update on project status | Priority: Medium  
Email: Team lunch invitation | Priority: Low  
Email: Newsletter subscription confirmation | Priority: Low  
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
PS D:\AI Coding>
```
- Sidebar:** Includes sections for "OUTLINE" and "TIMELINE".

## 2c) Perform classification using One-shot prompting

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows a folder named "AI CODING" containing files like "add.py", "AI lab43.py", "Assignment1(CP).pdf", etc.
- Code Editor:** Displays a Python file "lab assignment 44.py" with the following code:

```
8 #  
9 #  
10 #  
11 ======2c  
12 # Example email and priority  
13 email = "Please review the attached report when free"  
14 priority = "Normal" # Set the priority level
```
- Terminal:** Shows the command line output:

```
ModuleNotFoundError: No module named 'pandas'  
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
Email: Server is down and business operations stopped | Priority: High  
Email: Client meeting scheduled for tomorrow | Priority: High  
Email: Please review the attached report when free | Priority: Medium  
Email: Need update on project status | Priority: Medium  
Email: Team lunch invitation | Priority: Low  
Email: Newsletter subscription confirmation | Priority: Low  
PS D:\AI Coding>
```
- Sidebar:** Includes sections for "OUTLINE" and "TIMELINE".

## 2d) Perform classification using Few-shot prompting.

```

File Edit Selection View Go Run Terminal Help < > Q AI Coding
EXPLORER Welcome assignment 3.py AI lab43.py lab assignment 44.py
AI CODING
  add.py
  AI lab43.py
  Assignment1(CP).pdf
  Assignment 2-4.pdf
  assignment 3.4
  assignment 3.py
  Assignment2.pdf
  assignment3.4.docx
  factorial.py
  jobs.py
  jobspp.py
  lab assignment 44.py
  lab assignment3.3.pdf
  lab1 HCP.pdf
  matrixHCP.py
  Untitled20.ipynb
  week2 HCP.pdf

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: "Client meeting scheduled for tomorrow"
Priority: Low
PS D:\AI Coding>

```

## 2e) Evaluate which technique produces the most reliable results and why.

```

File Edit Selection View Go Run Terminal Help < > Q AI Coding
EXPLORER Welcome assignment 3.py AI lab43.py lab assignment 44.py
AI CODING
  add.py
  AI lab43.py
  Assignment 2-4.pdf
  Assignment 3.4
  assignment 3.py
  Assignment2.pdf
  assignment3.4.docx
  factorial.py
  jobs.py
  jobspp.py
  lab assignment 44.py
  lab assignment3.3.pdf
  lab1 HCP.pdf
  matrixHCP.py
  Untitled20.ipynb
  week2 HCP.pdf

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Email: Team lunch invitation | Priority: Low
Email: Client meeting scheduled for tomorrow | Priority: High
Email: Please review the attached report when free | Priority: Medium
Email: Need update on project status | Priority: Medium
Email: Team lunch invitation | Priority: Low
Email: Newsletter subscription confirmation | Priority: Low
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Email: "Client meeting scheduled for tomorrow"
Priority: Low
PS D:\AI Coding>

```

## 3. Student Query Routing System

**Scenario:**

**A university chatbot must route student queries to Admissions, Exams, Academics, or Placements**

**3a) . Create 6 sample student queries mapped to departments.**

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

- File Explorer:** Shows a folder named "AI CODING" containing files like "add.py", "AI lab43.py", "Assignment (CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobscpy.py", "lab assignment ...", "lab assignment3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.ipynb", and "week2 HCP.pdf".
- Code Editor:** The active file is "lab assignment 44.py". It contains the following Python code:

```
student_queries = {  
    "How do I register for courses?": "Registrar",  
    "What is my current GPA?": "Admissions",  
    "I need to pay my tuition": "Finance",  
    "Can I get a transcript?": "Registrar",  
    "I'm having trouble with my financial aid": "Finance",  
    "How do I declare a major?": "Admissions"  
}  
  
for query, department in student_queries.items():  
    print(f"Query: {query}")  
    print(f"Department: {department}\n")
```
- Terminal:** Below the code editor, the terminal tab is selected. It shows the output of running the script:

```
Query: How do I register for courses?  
Department: Registrar  
  
Query: What is my current GPA?  
Department: Admissions  
  
Query: I need to pay my tuition  
Department: Finance  
  
Query: Can I get a transcript?  
Department: Registrar
```
- Sidebar:** Includes icons for user profile, settings, outline, and timeline.

**3b) Implement Zero-shot intent classification using an LLM.**

The screenshot shows the Visual Studio Code interface with the "AI Coding" extension active. The Explorer sidebar on the left lists files including "assignment 3.py", "AI lab43.py", "lab assignment 44.py", and "Assignment 1(CP).pdf". The "lab assignment 44.py" file is open in the main editor area, containing the following Python code:

```
1 #
2 if department == "Registrar":
3     classification = "Academics"
4 elif department == "Admissions":
5     classification = "Admissions"
6 elif department == "Finance":
7     classification = "Placements"
8 else:
9     classification = "Exams"
10 print("Classification: " + classification)
```

The terminal below the editor shows two interactions with the code:

```
Query: I'm having trouble with my financial aid
Department: Finance
Classification: Placements

Query: How do I declare a major?
Department: Admissions
Classification: Admissions
```

The status bar at the bottom indicates the path "PS D:\AI Coding".

**3c) Improve results using One-shot prompting.**

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

- File Explorer:** Shows files in the "AI CODING" folder, including "add.py", "AI lab43.pdf", "Assignment 1(CP).pdf", "Assignment 2-4.pdf", "assignment 3.4", "assignment 3.py", "Assignment2.pdf", "assignment3.4.docx", "factorial.py", "jobs.py", "jobsqp.py", "lab assignment ... (1).pdf", "lab assignment 3.3.pdf", "lab1 HCP.pdf", "matrixHCP.py", "Untitled20.ipynb", and "week2 HCP.pdf".
- Code Editor:** Displays the content of "lab assignment 44.py". The code defines a function `classify\_query(query)` that classifies a query to determine the appropriate department based on keyword matches against predefined lists for Exams, Academics, Admissions, Library, and Finance.
- Terminal:** Shows the output of running the script with various test queries, such as "When will results be announced?", "Explain syllabus for Data Structures", "How do I apply for admission?", and "Where can I borrow books?". The output indicates the department for each query: Admissions, Exams, Academics, and Academics respectively.
- Status Bar:** Shows the current file path as "D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe", the command "d:/AI Coding/lab assignment 44.py", and the prompt "Query: "When will results be announced?"

**3d) Further refine results using Few-shot prompting.**

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

- File Explorer (Left):** Shows a folder named "AI CODING" containing files like add.py, AI lab43.py, Assignment1(CP).pdf, Assignment 2-4.pdf, assignment 3.4, assignment 3.py, Assignment2.pdf, assignment3.4.docx, factorial.py, jobs.py, jobscp.py, and lab assignment ... (with a red error icon). Other files listed include lab assignment3.3.pdf, lab1 HCP.pdf, matrixHCP.py, Untitled20.ipynb, and week2 HCP.pdf.
- Code Editor (Center):** Displays a Python script named "lab assignment 44.py". The code defines a function `classify_query` that takes a query string and returns the department name based on keywords. It includes test cases for Admissions, Exams, and Placements.
- Terminal (Bottom):** Shows a command-line session where the script is run. It starts with "PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe d:/AI Coding/lab assignment 44.py". A `ModuleNotFoundError` is caught, printing "Admissions", "Exams", "Placements", "Admissions", and "PS D:\AI Coding>".

**3e) Analyze how contextual examples affect classification accuracy.**

The screenshot shows the Visual Studio Code (VS Code) interface with the "AI Coding" extension installed. The "TERMINAL" tab is active, displaying Python code for a "ClassificationAnalyzer" class. The code includes methods for zero-shot and one-shot classification prompts, along with example usage. Below the terminal, the "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "PORTS" tabs are visible. A status bar at the bottom shows "PS D:\AI Coding".

```
from collections import defaultdict
import json

class ClassificationAnalyzer:
    def __init__(self):
        self.results = defaultdict(list)

    def zero_shot_prompt(self, text, categories):
        """
        Zero-shot: No examples provided
        """
        prompt = f"""Classify the following text into one of these categories: {', '.join(categories)}\nText: {text}\nCategory:"""

    def one_shot_prompt(self, text, categories, example_text, example_category):
        """
        One-shot: Single example provided
        """
        prompt = f"""Classify text into categories: {', '.join(categories)}\nExample:\nText: {example_text}\nCategory: {example_category}"""

Ambiguity: Reduced - one example clarifies intent
Consistency: Improved - example sets pattern
Pros: Minimal overhead, Some context
Cons: Limited learning from one example

FEW_SHOT:
Accuracy: Higher - multiple references provided
Ambiguity: Significantly reduced - pattern clear
Consistency: High - multiple examples establish standard
Pros: Best accuracy, Clear patterns, Reduced errors
Cons: Requires manual examples, Prompt size
```

**4) Chatbot Question Type Detection Scenario:**

A chatbot must identify whether a user query is **Informational, Transactional, Complaint, or Feedback**.

**4a) Prepare 6 chatbot queries mapped to question types.**

```

def classify_query(query):
    """
    Classify a query into one of four types:
    - Informational
    - Transactional
    - Complaint
    - Feedback
    """
    query_lower = query.lower()

    # Define keywords for each category
    informational_keywords = ['what', 'how', 'when', 'where', 'why', 'working hours', 'reset', 'password']
    transactional_keywords = ['book', 'buy', 'order', 'purchase', 'reserve', 'ticket']
    complaint_keywords = ['hasn\'t', 'didn\'t', 'broken', 'bad', 'poor', 'issue', 'problem', 'arrived']
    feedback_keywords = ['great', 'good', 'excellent', 'user-friendly', 'nice', 'love', 'hate', 'experience']

    # Check for complaint (highest priority)
    if any(keyword in query_lower for keyword in complaint_keywords):
        return 'Complaint'

    # Check for transactional
    if any(keyword in query_lower for keyword in transactional_keywords):
        return 'Transactional'

    # Check for feedback
    if any(keyword in query_lower for keyword in feedback_keywords):
        return 'Feedback'

    # Informational
    return 'Informational'

```

Ambiguity: Significantly reduced - pattern clear  
Consistency: High - multiple examples establish standard  
Pros: Best accuracy, Clear patterns, Reduced errors  
Cons: Requires manual examples, Prompt size

PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
What are your working hours? | Informational  
I want to book a ticket | Transactional  
My order hasn't arrived yet | Complaint  
The app is very user-friendly | Feedback  
How can I reset my password? | Informational  
The service experience was bad | Complaint

#### 4b) Design prompts for Zero-shot, One-shot, and Few-shot learning.

```

# Query Classification System

def classify_query(query):
    """
    Classify a user query into one of four categories:
    - Informational: User seeking information
    - Transactional: User wanting to perform an action/transaction
    - Complaint: User expressing dissatisfaction
    - Feedback: User providing feedback or suggestions
    """

    query_lower = query.lower()

    # Transactional keywords
    transactional_keywords = ["book", "buy", "purchase", "order", "reserve", "checkout", "pay"]

    # Complaint keywords
    complaint_keywords = ["problem", "issue", "broken", "error", "complaint", "wrong", "not working"]

    # Feedback keywords
    feedback_keywords = ["feedback", "suggest", "idea", "improve", "opinion", "review"]

    # Informational keywords
    informational_keywords = ["how", "what", "when", "where", "why", "can you tell", "help", "information"]

    # Classify based on keywords
    if any(keyword in query_lower for keyword in transactional_keywords):
        return "Transactional"

    if any(keyword in query_lower for keyword in complaint_keywords):
        return "Complaint"

    if any(keyword in query_lower for keyword in feedback_keywords):
        return "Feedback"

    if any(keyword in query_lower for keyword in informational_keywords):
        return "Informational"

    return "Unknown"

```

Cons: Requires manual examples, Prompt size

PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
What are your working hours? | Informational  
I want to book a ticket | Transactional  
My order hasn't arrived yet | Complaint  
The app is very user-friendly | Feedback  
How can I reset my password? | Informational  
The service experience was bad | Complaint

PS D:\AI Coding & C:/Users/ANALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"  
Query: I want to book a ticket  
Type: Transactional

#### 4c) Test all prompts on the same unseen queries.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files including `assignment1(CP).pdf`, `Assignment 2-4.pdf`, `assignment 3.pdf`, `Assignment2.pdf`, `assignment3.4.docx`, `factorial.py`, `jobs.py`, `jobs.py`, `lab assignment ...`, `lab assignment3.3.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.ipynb`, and `week2 HCP.pdf`.
- Code Editor:** Displays a Python script named `lab assignment 44.py` with the following content:

```
def classify_feedback(query):
    """
    Classify Feedback into categories based on sentiment and content.

    Args:
        query (str): The feedback text to classify

    Returns:
        str: The classification type (Complaint, Compliment, Neutral, etc.)
    """
    # Define keywords for each category
    complaint_keywords = ['bad', 'poor', 'terrible', 'awful', 'horrible', 'worst', 'hates', 'issue', 'problem', 'broken']
    compliment_keywords = ['good', 'great', 'excellent', 'amazing', 'wonderful', 'friendly', 'love', 'best', 'perfect', 'awesome']

    # Check for complaints
    if any(keyword in query.lower() for keyword in complaint_keywords):
        return "Complaint"

    # Check for compliments
    if any(keyword in query.lower() for keyword in compliment_keywords):
        return "Compliment"

    # Default
    return "Neutral"
```

- Terminal:** Shows the following interaction:

```
How can I reset my password? | Informational
The service experience was bad | Complaint
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: I want to book a ticket
Type: Transactional
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: The service experience was bad
Type: Complaint
Query: "The app is very user-friendly"
Type: Compliment
PS D:\AI Coding>
```

#### 4d) Compare response correctness and ambiguity handling.

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files including `assignment1(CP).pdf`, `Assignment 2-4.pdf`, `assignment 3.pdf`, `Assignment2.pdf`, `assignment3.4.docx`, `factorial.py`, `jobs.py`, `jobs.py`, `lab assignment ...`, `lab assignment3.3.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.ipynb`, and `week2 HCP.pdf`.
- Code Editor:** Displays a Python script named `lab assignment 44.py` with the following content:

```
# This function classifies user queries into different types
def classify_query(query):
    if "book" in query.lower() or "reserve" in query.lower():
        return "Transactional"
    elif "order" in query.lower() or "arrived" in query.lower():
        return "Complaint"
    elif "support" in query.lower() or "great" in query.lower():
        return "Feedback"
    else:
        return "Informational"

# Example usage
query = "I want to book a ticket"
query_type = classify_query(query)
print(f"Query: {query}\nType: {query_type}")

# Another example
query = "The app is very user-friendly"
query_type = classify_query(query)
print(f"Query: {query}\nType: {query_type}")
```

- Terminal:** Shows the following interaction:

```
Query: I want to book a ticket
Type: Transactional
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: The service experience was bad
Type: Complaint
Query: "The app is very user-friendly"
Type: Compliment
PS D:\AI Coding> & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Query: I want to book a ticket
Type: Transactional
PS D:\AI Coding>
```

#### 4e) Document observations.

```

1 """
2 Observations from testing Zero-shot, One-shot, and Few-shot prompting for classification tasks:
3
4 1. Accuracy:
5     - Zero-shot prompting often resulted in lower accuracy compared to One-shot and Few-shot methods, as the model had no prior examples to reference.
6     - One-shot prompting showed improved accuracy, as the model could leverage a single example to understand the task better.
7     - Few-shot prompting consistently yielded the highest accuracy, as multiple examples provided the model with a clearer context and better understanding of the classification task.
8
9 2. Ambiguity Handling:
10    - Zero-shot prompting struggled with ambiguous inputs, often leading to incorrect classifications due to lack of context.
11    - One-shot prompting reduced ambiguity to some extent, as the provided example helped clarify the task, but some ambiguity remained.
12    - Few-shot prompting effectively handled ambiguity by providing multiple examples that illustrated different aspects of the classification task, allowing the model to make more informed decisions.
13
14 3. Consistency:
15    - Zero-shot prompting exhibited high variability in results, with performance heavily dependent on the specific input phrasing.
16    - One-shot prompting showed moderate consistency, as the single example could guide the model, but variations in input still affected outcomes.
17    - Few-shot prompting demonstrated the highest consistency across different inputs, as the multiple examples helped stabilize the model's responses.
18
19 4. Overall Performance Differences:
20    - Overall, Few-shot prompting outperformed both Zero-shot and One-shot methods in terms of accuracy, ambiguity handling, and consistency.
21    - Zero-shot prompting was useful for quick assessments but lacked reliability for critical tasks.
22    - One-shot prompting served as a middle ground, offering some improvements but still falling short of Few-shot performance.
23 """

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Type: Transactional  
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab\_assignment\_44.py"  
Query: "The service experience was bad"  
Type: Complaint

Query: "The app is very user-friendly"  
Type: Complaint  
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab\_assignment\_44.py"  
Query: 'I want to book a ticket'  
Type: Transactional  
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab\_assignment\_44.py"  
PS D:\AI Coding

## 5) Emotion Detection in Text

### Scenario:

A mental-health chatbot needs to detect emotions: Happy, Sad, Angry, Anxious, Neutral.

### 5a) Create labeled emotion samples.

```

1 import pandas as pd
2
3 # Create a DataFrame from the provided data
4 data = {
5     "Text": [
6         "I am very happy today",
7         "I feel lonely and depressed",
8         "This is so frustrating",
9         "I am worried about my future",
10        "Today is just normal",
11        "Feeling excited about results"
12    ],
13    "Emotion": [
14        "Happy",
15        "Sad",
16        "Angry",
17        "Anxious",
18        "Neutral",
19        "Happy"
20    ]
21 }
22
23 df = pd.DataFrame(data)
24
25 # Display the DataFrame
26 print(df)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab\_assignment\_44.py"  
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab\_assignment\_44.py"  
Traceback (most recent call last):  
 File "d:/AI Coding\lab\_assignment\_44.py", line 1, in <module>  
 import pandas as pd  
**ModuleNotFoundError: No module named 'pandas'**  
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab\_assignment\_44.py"  
Traceback (most recent call last):  
 File "d:/AI Coding\lab\_assignment\_44.py", line 1, in <module>  
 import pandas as pd  
**ModuleNotFoundError: No module named 'pandas'**  
PS D:\AI Coding

**5b) Use Zero-shot prompting to identify emotions.**

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files including `assignment 3.py`, `AI lab43.py`, `Assignment 1(C).pdf`, `Assignment 2-4.pdf`, `assignment 3.4`, `assignment 3.py`, `Assignment2.pdf`, `assignment3.docx`, `factorial.py`, `jobs.py`, `jobscc.py`, `lab assignment ...`, `lab assignment33.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.py`, and `week2 HCP.pdf`.
- Code Editor:** Displays a Python script named `lab assignment 44.py` with the following code:

```
def identify_emotion(text):
    if "worried" in text:
        return "Anxious"
    return "Neutral"

text = "I am worried about my future"
emotion = identify_emotion(text)
print(f"Emotion: {emotion}")
```
- Terminal:** Shows the following command-line output:

```
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:\AI Coding\lab assignment 44.py"
Traceback (most recent call last):
  File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:\AI Coding\lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding>
```

**5c) Use One-shot prompting with an example.**

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files including `assignment 3.py`, `AI lab43.py`, `Assignment 1(C).pdf`, `Assignment 2-4.pdf`, `assignment 3.4`, `assignment 3.py`, `Assignment2.pdf`, `assignment3.docx`, `factorial.py`, `jobs.py`, `jobscc.py`, `lab assignment ...`, `lab assignment33.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.py`, and `week2 HCP.pdf`.
- Code Editor:** Displays a Python script named `lab assignment 44.py` with the following code:

```
def identify_emotion(text):
    if "frustrating" in text:
        return "Frustrated"
    return "Neutral"

# Example usage
text = "This is so frustrating"
emotion = identify_emotion(text)
print(f"Emotion: {emotion}")
```
- Terminal:** Shows the following command-line output:

```
File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:\AI Coding\lab assignment 44.py"
Traceback (most recent call last):
  File "d:\AI Coding\lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:\AI Coding\lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding>
```

**5d) Use Few-shot prompting with multiple emotions.**

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows files like `add.py`, `AI lab43.pdf`, `Assignment 1(CP).pdf`, `Assignment 2-4.pdf`, `assignment 3.4`, `assignment 3.py`, `Assignment2.pdf`, `assignment3.4.docx`, `factorial.py`, `jobs.py`, `jobsccp.py`, `lab assignment ...`, `lab assignment3.3.pdf`, `lab1 HCP.pdf`, `matrixHCP.py`, `Untitled20.ipynb`, and `week2 HCP.pdf`.
- Editor:** Displays Python code for emotion classification. The code defines a function `classify_emotion` that takes a text string and returns its emotion based on a dictionary of keywords.
- Terminal:** Shows the following output from a terminal window:
 

```
Traceback (most recent call last):
  File "d:/AI Coding/lab assignment 44.py", line 1, in <module>
    import pandas as pd
ModuleNotFoundError: No module named 'pandas'
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: "This is so frustrating"
Emotion: frustrated
PS D:\AI Coding
```

## 5e) Discuss ambiguity handling across techniques.

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows files similar to the previous screenshot.
- Editor:** Displays Python code for handling ambiguity in emotion recognition. It defines a function `handle_emotion` that takes a technique name and an input text, returning a descriptive message about the technique's performance.
- Terminal:** Shows the following output from a terminal window:
 

```
# Emotion Handling Techniques
def handle_emotion(technique, input_text):
    if technique == "zero-shot":
        return "This technique struggles with ambiguity in understanding emotions."
    elif technique == "one-shot":
        return "This technique provides better clarity in emotional interpretation."
    elif technique == "few-shot":
        return "This technique achieves the best emotional accuracy by learning from examples."
    else:
        return "Unknown technique."

# Example usage
Techniques = ["zero-shot", "one-shot", "few-shot"]
for technique in techniques:
    print(f"[{technique.capitalize()}]: {handle_emotion(technique, '')}")
```

```
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Anxious
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Emotion: Frustrated
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Text: "This is so frustrating"
Emotion: frustrated
PS D:\AI Coding & C:/Users/ANJALI/AppData/Local/Programs/Python/Python313/python.exe "d:/AI Coding/lab assignment 44.py"
Zero-shot: This technique struggles with ambiguity in understanding emotions.
One-shot: This technique provides better clarity in emotional interpretation.
Few-shot: This technique achieves the best emotional accuracy by learning from examples.
PS D:\AI Coding
```