

ASSIGNMENT – 4.5

NAME:SD SUFIYAN

NO:2303A51980

BATCH NO:30

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: 3.1.PY, 4.2.PY, and 4.5.PY. The 4.5.PY tab is active, displaying the following Python code:

```
1 # Sample data
2 test_emails = [
3     "My payment failed but money was deducted.",
4     "The app is not opening on my phone.",
5     "Great customer service, very satisfied."
6     "What is your customer care number?",
7     "Invoice amount seems incorrect."
8 ]
9 # Expected labels (for evaluation)
10 true_labels = [
11     "Billing",
12     "Technical support",
13     "Feedback",
14     "Others",
15     "Billing"
16 ]
17 # Simple rule-based classifier
18 def classify_email(email):
19     email = email.lower()
20
21     if any(word in email for word in ["payment", "invoice", "amount", "deducted", "bill"]):
22         return "Billing"
23
24     Email: The app is not opening on my phone.
25     Predicted Category: Technical Support
26
27     Email: Great customer service, very satisfied.
28     Predicted Category: Feedback
```

The code defines a function `classify_email` that takes an email string and returns its predicted category based on a simple rule set. It also includes three sample emails and their predicted categories.

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:

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.

DECOMPT: Email "How can I obtain a copy of my Bill?"

■ **Methodology** and **Results** ■ **Conclusion**

Email: "I love the new update." → Feedback

Email: "What are your office hours?" → Other

```

OPEN EDITORS
  4.5.py > ...
  4.5.py
  4.5.py

SESSIONS
  SESSION 1
  SESSION 2
  SESSION 3
  SESSION 4
  SESSION 5
  SESSION 6
  SESSION 7
  SESSION 8
  SESSION 9
  SESSION 10
  SESSION 11
  SESSION 12
  SESSION 13
  SESSION 14
  SESSION 15
  SESSION 16
  SESSION 17
  SESSION 18
  SESSION 19
  SESSION 20
  SESSION 21
  SESSION 22
  SESSION 23
  SESSION 24
  SESSION 25
  SESSION 26
  SESSION 27
  SESSION 28
  SESSION 29
  SESSION 30
  SESSION 31
  SESSION 32
  SESSION 33
  SESSION 34
  SESSION 35
  SESSION 36
  SESSION 37
  SESSION 38
  SESSION 39
  SESSION 40
  SESSION 41
  SESSION 42
  SESSION 43
  SESSION 44
  SESSION 45
  SESSION 46
  SESSION 47
  SESSION 48
  SESSION 49
  SESSION 50
  SESSION 51
  SESSION 52
  SESSION 53
  SESSION 54
  SESSION 55
  SESSION 56
  SESSION 57
  SESSION 58
  SESSION 59
  SESSION 60
  SESSION 61
  SESSION 62
  SESSION 63
  SESSION 64
  SESSION 65
  SESSION 66
  SESSION 67
  SESSION 68
  SESSION 69
  SESSION 70
  SESSION 71
  SESSION 72
  SESSION 73
  SESSION 74
  SESSION 75
  SESSION 76
  SESSION 77
  SESSION 78
  SESSION 79
  SESSION 80
  SESSION 81
  SESSION 82
  SESSION 83
  SESSION 84
  SESSION 85
  SESSION 86
  SESSION 87
  SESSION 88
  SESSION 89
  SESSION 90
  SESSION 91
  SESSION 92
  SESSION 93
  SESSION 94
  SESSION 95
  SESSION 96
  SESSION 97
  SESSION 98
  SESSION 99
  SESSION 100
  SESSION 101
  SESSION 102
  SESSION 103
  SESSION 104
  SESSION 105
  SESSION 106
  SESSION 107
  SESSION 108
  SESSION 109
  SESSION 110
  SESSION 111
  SESSION 112
  SESSION 113
  SESSION 114
  SESSION 115

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRESQL QUERY RESULTS
syedsufiyan@syeds-MacBook-Air-3:4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py
Email: "The app is not opening on my phone."
Category: Technical Support
Email: "Great customer service, very satisfied."
Category: Feedback
Email: "What is your customer care number?"

```

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 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 several tabs open. The active tab is '4.5.py'. The code in the editor is as follows:

```
OPEN EDITORS
  4.5.py
  4.5.PY
  4.5.py

zero_shot_coding_classification(coding_queries)
# programming_prompts.py
def one_shot_coding_classification(queries):
    print("\n==== ONE-SHOT PROMPTING ====\n")
    for query in queries:
        print(f"Prompt sent to AI with one example:\nQuery: '{query}'\n")
        q = query.lower()
        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"
        print("AI Output:", output)
        print("-" * 40)
coding_queries = [
    "Why am I getting IndexError in my Python list?",
    "My sorting algorithm is 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?"
]
one_shot_coding_classification(coding_queries)
#
few_shot_coding_classification(queries):
    print("\n==== FEW-SHOT PROMPTING ====\n")
    for query in queries:
        print(f"Prompt sent to AI with multiple examples:\nQuery: '{query}'\n")
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRES QUERY RESULTS
syedsufiyan@syeds-MacBook-Air-3:4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py
Query: "Cancel my hotel reservation in Paris."
AI Output: Cancellation
Query: "What is the baggage allowance?"
AI Output: General Travel Info
```

The terminal window at the bottom shows the output of running the script. It correctly identifies the first query as 'Cancellation' and the second as 'General Travel Info'.

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 multiple tabs open. The active tab is '4.5.py', which contains Python code for classifying AI queries. The code uses dictionaries to map error messages to output responses like 'Syntax Error' or 'Logic Error'. It also includes functions for zero-shot and one-shot classification. Below the code editor, there are tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and 'POSTGRESQL QUERY RESULTS'. The terminal window shows command-line interactions with the AI system, including queries about flight bookings, hotel reservations, and baggage allowances. A status bar at the bottom indicates that the user has reached the limit for chat messages.

```
def zero_shot_coding_classification(queries):
    print("\n==== ZERO-SHOT PROMPTING ===\n")
    for query in queries:
        print(f"\nQuery: \"{query}\"\n")
        q = query.lower()
        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"
        print("AI Output:", output)
    print("-" * 40)
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?"
]
zero_shot_coding_classification(coding_queries)
# programming_prompts.py
def one_shot_coding_classification(queries):
    print("\n==== ONE-SHOT PROMPTING ===\n")
    for query in queries:
        print(f"\nQuery sent to AI with one example:\nQuery: \"{query}\"\n")
        q = query.lower()
        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        else:
            output = "Conceptual Question"
    print("AI Output: " + output)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRESQL QUERY RESULTS

syedsufiyanSyeds-MacBook-Air-3:4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py

Query: "Book a flight from Delhi to Mumbai."

AI Output: Flight Booking

Query: "Cancel my hotel reservation in Paris."

AI Output: Cancellation

Query: "What is the baggage allowance?"

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 Jupyter Notebook interface with a dark theme. On the left, the 'OPEN EDITORS' sidebar lists three files: '4.5.py' (the active file), '4.5.PY', and '4.5.py'. The main area displays a Python script named '4.5.py' with the following code:

```
def few_shot_travel_classification(queries):
    print("\n==== FEW-SHOT PROMPTING ===\n")
    for query in queries:
        print(f'Query: "{query}"')
        if "cancel" in query.lower():
            output = "Cancellation"
        elif "flight" in query.lower():
            output = "Flight Booking"
        elif "hotel" in query.lower():
            output = "Hotel Booking"
        else:
            output = "General Travel Info"
        print("AI Output:", output)
        print("-" * 40)
    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."
    ]
    few_shot_travel_classification(travel_queries)

# programming_prompts.py
def zero_shot_coding_classification(queries):
    print("\n==== ZERO-SHOT PROMPTING ===\n")
    for query in queries:
        print(f"Query: {query}")
        print("AI Output:", output)
        print("-" * 40)
```

The terminal below shows the execution of the script:

```
syedsufiyan@Syeds-MacBook-Air-3: ~ 4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py
Query: "Book a flight from Delhi to Mumbai."
AI Output: Flight Booking
Query: "Cancel my hotel reservation in Paris."
AI Output: Cancellation
Query: "What is the baggage allowance?"
AI Output: General Travel Info
```

The right side of the interface shows a 'SESSIONS' panel with a single session named '4.5.py'.

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:

Observation (Zero-shot Prompting)

- Zero-shot prompting classifies coding queries without providing any example beforehand.
 - The classification is based only on keywords present in the query.
 - Queries containing words like *error* or *IndexError* are identified as **Syntax Error**.
 - Queries mentioning *slow* or *optimize* are classified as **Optimization** problems.
 - Queries related to incorrect outputs are categorized as **Logic Error**.
 - Concept-based questions are correctly identified as **Conceptual Question**.
 - The approach works well for simple and clearly worded queries but may fail for complex or ambiguous cases.

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:

```
OPEN EDITORS
4.5.py
4.5.PY
4.5.py

def one_shot_coding_classification(queries):
    for query in queries:
        print(f'Prompt sent to AI with one example:\nQuery: "{query}"\n')
        q = query.lower()
        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"
        print("AI Output:", output)
        print("-" * 40)
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?"
]
one_shot_coding_classification(coding_queries)
#
def few_shot_coding_classification(queries):
    print("\n==== FEW-SHOT PROMPTING ===\n")
    for query in queries:
        print(f'Prompt sent to AI with one example:\nQuery: "{query}"\n')
        q = query.lower()
        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"
        print("AI Output:", output)
        print("-" * 40)
syedsufiyan@syedsufiyan-MacBook-Air-3:~/Documents$ python3 4.5.py
Query: "Explain the difference between list and tuple in Python."
AI Output: Conceptual Question
Query: "How can I optimize my recursive Fibonacci function?"
AI Output: Optimization
```

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 dark-themed VS Code interface. On the left is a sidebar with icons for file operations like Open, Save, Find, and Delete. The top navigation bar includes 'OPEN EDITORS' (with '4.5.py' and '4.5.PY' listed), 'SESSIONS' (with a search icon), and file tabs for '4.5.py' and '4.5.PY'. The main editor area contains a Python script with code for few-shot and zero-shot AI classification. The script uses print statements to interact with the AI. The bottom status bar shows the command line: 'syedsufiyan@syedsufiyan-MacBook-Air-3: ~ % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py'. The bottom right corner has a message: 'You've reached the limit for chat... Upgrade'.

```
OPEN EDITORS
SESSIONS
4.5.py
4.5.PY

4.5.py ... 4.5.py

def few_shot_coding_classification(queries):
    print("\n==== FEW-SHOT PROMPTING ====\n")

    for query in queries:
        print(f'Prompt sent to AI with multiple examples:\nQuery: "{query}"\n')

        q = query.lower()

        if "error" in q or "indexerror" in q:
            output = "Syntax Error"
        elif "wrong" in q:
            output = "Logic Error"
        elif "slow" in q or "optimize" in q:
            output = "Optimization"
        else:
            output = "Conceptual Question"

        print("AI Output:", output)
        print("-" * 40)

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?"
]

few_shot_coding_classification(coding_queries)
# def zero_shot_social_classification(domains):
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ...
Python + - x
syedsufiyan@syedsufiyan-MacBook-Air-3: ~ % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py
AI Output: Logic Error
Prompt sent to AI with one example:
Query: "Explain the difference between list and tuple in Python."
AI Output: Conceptual Question
Prompt sent to AI with one example:
Describe what to build next
```

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:

```

OPEN EDITORS
4.5.py > ...
4.5.PY
4.5.py

4.5.py > ...
186 def zero_shot_social_classification(posts):
187     for post in posts:
188         print(f"Post: '{post}'")
189
190         p = post.lower()
191
192         if "disappointed" in p or "not arrived" in p or "bad" in p:
193             output = "Complaint"
194         elif "discount" in p or "offer" in p or "sale" in p:
195             output = "Promotion"
196         elif "thank" in p or "great" in p or "support" in p:
197             output = "Appreciation"
198         else:
199             output = "Inquiry"
200
201         print("AI Output:", output)
202         print("-" * 40)
203
204 social_posts = [
205     "My order has not arrived yet, very disappointed.",
206     "Get 50% discount on all products today!",
207     "Thanks for the quick customer support!",
208     "Can you tell me your return policy?"
209 ]
210
211 zero_shot_social_classification(social_posts)
212 #
213 # One-shot social media post classification
214 #
215 #
216 # Few-shot social media post classification
217 def few_shot_social_classification(posts):
218     predictions = []

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ...

syedsufiyan@Syeds-MacBook-Air-3: ~ 4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py

==== ZERO-SHOT PROMPTING ===

Post: "My order has not arrived yet, very disappointed."
AI Output: Complaint

Post: "Get 50% discount on all products today!"
AI Output: Promotion

SESSIONS

You've reached the limit for chat ... Upgrade

+ 4.5.py

Describe what to build next

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>"

The screenshot shows a Visual Studio Code interface. On the left, there's a sidebar with icons for search, file, folder, and others. The top navigation bar has tabs for 'OPEN EDITORS' (with '4.5.py' selected), 'SESSIONS', and other icons. The main area has two panes: one for code and one for a terminal.

Code Editor Content:

```
4.5.py > ...
212     zero_shot_social_classification(social_posts)
213 #
214 # One-shot social media post classification
215
216 #
217 def few_shot_social_classification(posts):
218     predictions = []
219
220     for post in posts:
221         prompt = (
222             "Examples:\n"
223             "Post: \"Loved the new feature!\" -> Appreciation\n"
224             "Post: \"My order hasn't arrived.\" -> Complaint\n"
225             "Post: \"Check out our discount offer!\" -> Promotion\n"
226             "Post: \"How can I reset my password?\" -> Inquiry\n\n"
227             "Now classify the following social media post:\n"
228             f"Post: \'{post}\'"
229         )
230
231         print("\nPROMPT SENT TO MODEL:\n")
232         print(prompt)
233         print("\nMODEL OUTPUT:")
234
235         # Simulated model reasoning using simple logic
236         text = post.lower()
237
238         if "love" in text or "great" in text or "thanks" in text or "happy" in text:
239             label = "Appreciation"
240         elif "not arrived" in text or "issue" in text or "wrong" in text or "problem"
241             label = "Complaint"
242         elif "discount" in text or "offer" in text or "sale" in text:
243             label = "Promotion"
```

Terminal Content:

```
syedsufiyan@MacBook-Air-3 4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py
```

Prompt sent to AI with multiple examples:
Query: "Explain the difference between list and tuple in Python."

AI Output: Conceptual Question

Prompt sent to AI with multiple examples:
Query: "How can I optimize my recursive Fibonacci function?"

Bottom Right Panel:

You've reached the limit for chat ... [Upgrade](#)

+ 4.5.py

Describe what to build next

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>"

The screenshot shows a code editor interface with a dark theme. On the left, there's a sidebar with icons for file operations like Open Editors, Save, Find, and Undo. The main area displays a Python script named `4.5.py`. The code defines a function `def few_shot_social_classification(posts)` that takes a list of posts as input. It prints examples of posts and their corresponding labels (Appreciation, Complaint, Promotion, Inquiry, Others). It then prints a prompt for the user to send to the model. The script uses simple logic to classify posts based on keywords like "love", "great", "thanks", "happy", "arrived", "issue", "wrong", "problem", "discount", "offer", "sale", "help", "how", "password", and "login". If none of these keywords are found, it classifies the post as "Others". Finally, it prints the predicted label and a separator line. The code ends with a note about few-shot learning and imports for problems, output, debug console, terminal, ports, and PostgreSQL query results.

```
OPEN EDITORS
4.5.py
4.5.py

def few_shot_social_classification(posts):
    """
    Post: "I loved the new feature!" -> Appreciation\n"
    Post: "My order hasn't arrived." -> Complaint\n"
    Post: "Check out our discount offer!" -> Promotion\n"
    Post: "How can I reset my password?" -> Inquiry\n\n"
    Now classify the following social media post:\n"
    Post: "(post)\n"
    )

    print("\nPROMPT SENT TO MODEL:\n")
    print(prompt)
    print("\nMODEL OUTPUT:")

    # Simulated model reasoning using simple logic
    text = post.lower()

    if "love" in text or "great" in text or "thanks" in text or "happy" in text:
        label = "Appreciation"
    elif "not arrived" in text or "issue" in text or "wrong" in text or "problem" in text:
        label = "Complaint"
    elif "discount" in text or "offer" in text or "sale" in text:
        label = "Promotion"
    elif "help" in text or "how" in text or "password" in text or "login" in text:
        label = "Inquiry"
    else:
        label = "Others"

    print(label)
    print("-" * 40)

    predictions.append(label)

    return predictions

# Few-shot learning
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTGRESQL QUERY RESULTS
syedsufiyan@Syeds-MacBook-Air-3:~/Documents$ python3 4.5.py % /usr/local/bin/python3 /Users/syedsufiyan/4.5.py/4.5.py
PROMPT SENT TO MODEL:

Examples:
Post: "I 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: "My order hasn't arrived yet."
MODEL OUTPUT:
Others

You've reached the limit for chat ...
Upgrade
```

OBSERVATION:

Provides multiple examples showing patterns to AI.

Highest accuracy; better handles informal, slang, or mixedlanguage posts.
Slightly longer prompts but most reliable for social media classification.