

Lab Assignment- 4.1

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

AI Assisted Coding

Navya Revuri– 2303A52483

Problem Statement 1: Customer Email Classification



The screenshot shows a Google Colab notebook titled "Lab_Assignment_4.1_2256.ipynb". The code is written in Python and defines a list of sample customer emails with their corresponding categories. It also includes a function to run prompts and a zero-shot prompt for classification.

```
[1] ✓ On
# Sample customer emails
sample_emails = [
    ("I was charged twice for my subscription this month.", "Billing"),
    ("My application crashes whenever I open it.", "Technical Support"),
    ("I really like the new update. Great job!", "Feedback"),
    ("What are your customer support working hours?", "Others"),
    ("I am unable to login to my account.", "Technical Support")
]

for email, category in sample_emails:
    print(f"Email: {email}")
    print(f"Category: {category}\n")

Email: I was charged twice for my subscription this month.
Category: Billing

Email: My application crashes whenever I open it.
Category: Technical Support

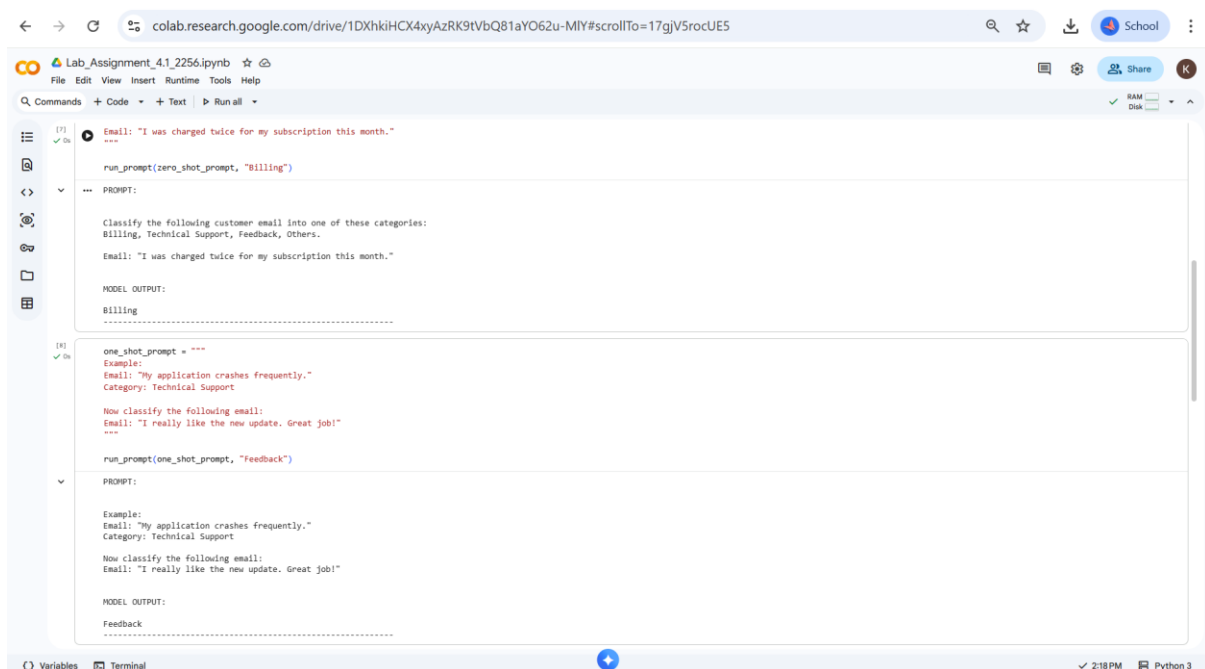
Email: I really like the new update. Great job!
Category: Feedback

Email: What are your customer support working hours?
Category: Others

Email: I am unable to login to my account.
Category: Technical Support

[4] ✓ On
def run_prompt(prompt, output):
    print(f"PROMPT: {prompt}")
    print(f"MODEL OUTPUT: {output}")
    print(f"Category: {category}\n")

[7] ✓ On
zero_shot_prompt = """
Classify the following customer email into one of these categories:
Billing, Technical Support, Feedback, Others.
Email: "I was charged twice for my subscription this month."
"""
```



The screenshot shows the same Google Colab notebook, but now it displays the output of the zero-shot and one-shot prompts. The zero-shot prompt correctly classifies the email as "Billing". The one-shot prompt correctly classifies the email as "Feedback".

```
[7] ✓ On
Email: "I was charged twice for my subscription this month."
"""

run_prompt(zero_shot_prompt, "Billing")

PROMPT:

Classify the following customer email into one of these categories:
Billing, Technical Support, Feedback, Others.
Email: "I was charged twice for my subscription this month."

MODEL OUTPUT:
Billing

[8] ✓ On
one_shot_prompt = """
Example:
Email: "My application crashes frequently."
Category: Technical Support

Now classify the following email:
Email: "I really like the new update. Great job!"
"""

run_prompt(one_shot_prompt, "Feedback")

PROMPT:

Example:
Email: "My application crashes frequently."
Category: Technical Support

Now classify the following email:
Email: "I really like the new update. Great job!"

MODEL OUTPUT:
Feedback
```

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
feedback
-----

[9] ✓ On
few_shot_prompt = """
Example 1:
Email: "I was charged twice for my subscription."
Category: Billing

Example 2:
Email: "My app crashes after the update."
Category: Technical Support

Example 3:
Email: "Great service and fast response."
Category: Feedback

Now classify:
Email: "What are your customer support working hours?"
"""

run_prompt(few_shot_prompt, "Others")

--- PROMPT:

Example 1:
Email: "I was charged twice for my subscription."
Category: Billing

Example 2:
Email: "My app crashes after the update."
Category: Technical Support

Example 3:
Email: "Great service and fast response."
Category: Feedback

Now classify:
Email: "What are your customer support working hours?"

MODEL OUTPUT:

Others
-----

Final Observation



- Zero-shot works for simple cases
- One-shot improves clarity
- Few-shot provides the best accuracy

```

Problem Statement 2: Intent Classification for Chatbot Queries

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

RAM Disk

```
[11] # Problem Statement-2
# Sample chatbot user queries with their intents

sample_queries = [
    ("I cannot access my account.", "Account Issue"),
    ("Where is my order now?", "Order Status"),
    ("Does this phone support 5G?", "Product Inquiry"),
    ("What are your working hours?", "General Question"),
    ("My password reset link is not working.", "Account Issue"),
    ("When will my package be delivered?", "Order Status")
]

for query, intent in sample_queries:
    print(f"Query: {query}")
    print(f"Intent: {intent}\n")

--- Query: I cannot access my account.
Intent: Account Issue

Query: Where is my order now?
Intent: Order Status

Query: Does this phone support 5G?
Intent: Product Inquiry

Query: What are your working hours?
Intent: General Question

Query: My password reset link is not working.
Intent: Account Issue

Query: When will my package be delivered?
Intent: Order Status

[12] ✓ On
def run_prompt(prompt, output):
    print("PROMPT:\n")
    print(prompt)
    print("\nMODEL OUTPUT:\n")
    print(output)
    print("-" * 60)

[13] ✓ On
zero_shot_prompt = """
Classify the following user query into one of these intents:
Account Issue, Order Status, Product Inquiry, General Question.

Query: "Where is my order now?"
"""

run_prompt(zero_shot_prompt, "Order Status")

--- PROMPT:

Classify the following user query into one of these intents:
Account Issue, Order Status, Product Inquiry, General Question.

Query: "Where is my order now?"
```

Variables Terminal

2:46 PM Python 3

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands Code Text Run all

RAM Disk

Query: "where is my order now?"

MODEL OUTPUT:

Order Status

.....

[14] ✓ On

one_shot_prompt = ""

Example:

Query: "I cannot log into my account."

Intent: Account Issue

Now classify the following query:

Query: "Does this phone support wireless charging?"

.....

run_prompt(one_shot_prompt, "Product Inquiry")

PROMPT:

Example:

Query: "I cannot log into my account."

Intent: Account Issue

Now classify the following query:

Query: "Does this phone support wireless charging?"

MODEL OUTPUT:

Product Inquiry

.....

[15] ✓ On

few_shot_prompt = ""

Example 1:

Query: "I forgot my password."

Intent: Account Issue

Example 2:

Query: "When will my package be delivered?"

Intent: Order Status

Example 3:

Query: "Is this laptop good for gaming?"

Intent: Product Inquiry

Now classify the following query:

Query: "What time does customer support open?"

.....

run_prompt(few_shot_prompt, "General Question")

PROMPT:

.....

Variables Terminal 2:46 PM Python 3

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands Code Text Run all

RAM Disk

[15] ✓ On

Example 2:

Query: "When will my package be delivered?"

Intent: Order Status

Example 3:

Query: "Is this laptop good for gaming?"

Intent: Product Inquiry

Now classify the following query:

Query: "What time does customer support open?"

.....

run_prompt(few_shot_prompt, "General Question")

PROMPT:

Example 1:

Query: "I forgot my password."

Intent: Account Issue

Example 2:

Query: "When will my package be delivered?"

Intent: Order Status

Example 3:

Query: "Is this laptop good for gaming?"

Intent: Product Inquiry

Now classify the following query:

Query: "What time does customer support open?"

MODEL OUTPUT:

General Question

.....

[16] ✓ On

print("Evaluation Summary:")

print("Zero-shot Output : Order Status")

print("One-shot Output : Product Inquiry")

print("Few-shot Output : General Question")

Evaluation Summary:

Zero-shot Output : Order Status

One-shot Output : Product Inquiry

Few-shot Output : General Question

Observation

Zero-shot prompting correctly identifies clear intents but may lack precision for ambiguous queries.

One-shot prompting improves intent clarity by providing a reference example.

Few-shot prompting gives the most accurate and reliable classification due to multiple contextual examples.

[]

Start coding or generate with AI.

Variables Terminal 2:46 PM Python 3

Problem Statement 3: Student Feedback Analysis

```
Lab_Assignment_4.1.2256.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text + Runall
RAM
Disk

[17]
# Problem statement -3
# Sample student feedback with sentiment labels

sample_feedback = [
    ("The course content was very informative.", "Positive"),
    ("The lectures were boring and unclear.", "Negative"),
    ("Classes were conducted regularly.", "Neutral"),
    ("The instructor explained concepts clearly.", "Positive"),
    ("The syllabus is outdated.", "Negative")
]

for feedback, sentiment in sample_feedback:
    print(f"Feedback: {feedback}")
    print(f"Sentiment: {sentiment}\n")

Feedback: The course content was very informative.
Sentiment: Positive

Feedback: The lectures were boring and unclear.
Sentiment: Negative

Feedback: Classes were conducted regularly.
Sentiment: Neutral

Feedback: The instructor explained concepts clearly.
Sentiment: Positive

Feedback: The syllabus is outdated.
Sentiment: Negative

[18]
def run_prompt(prompt, output):
    print("PROMPT:\n")
    print(prompt)
    print("\nMODEL OUTPUT:\n")
    print(output)
    print("." * 60)

[20]
zero_shot_prompt = """
Classify the following student feedback as:
Positive, Negative, or Neutral.

Feedback: "The course content was very informative."
"""

run_prompt(zero_shot_prompt, "Positive")

PROMPT:

Classify the following student feedback as:
Positive, Negative, or Neutral.

Feedback: "The course content was very informative."

MODEL OUTPUT:

Positive
.....
```

```
Lab_Assignment_4.1.2256.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text + Runall
RAM
Disk

MODEL OUTPUT:
Positive
.....

[22]
one_shot_prompt = """
Example:
Feedback: "The lectures were boring."
Sentiment: Negative

Now classify the following feedback:
Feedback: "The assignments were manageable."
"""

run_prompt(one_shot_prompt, "Neutral")

PROMPT:

Example:
Feedback: "The lectures were boring."
Sentiment: Negative

Now classify the following feedback:
Feedback: "The assignments were manageable."

MODEL OUTPUT:

Neutral
.....

[23]
few_shot_prompt = """
Example 1:
Feedback: "Excellent teaching methods."
Sentiment: Positive

Example 2:
Feedback: "The syllabus is outdated."
Sentiment: Negative

Example 3:
Feedback: "Classes were conducted regularly."
Sentiment: Neutral

Now classify the following feedback:
Feedback: "The instructor explained concepts clearly."
"""

run_prompt(few_shot_prompt, "Positive")

PROMPT:

Example 1:
Feedback: "Excellent teaching methods."
Sentiment: Positive
```

The screenshot shows a Jupyter Notebook titled 'Lab_Assignment_4.1_2256.ipynb'. It contains three examples of sentiment classification using a model. Example 1 shows 'Excellent teaching methods.' classified as 'Positive'. Example 2 shows 'The syllabus is outdated.' classified as 'Negative'. Example 3 shows 'Classes were conducted regularly.' classified as 'Neutral'. A fourth example shows 'The instructor explained concepts clearly.' classified as 'Positive'. Below the examples, an 'Evaluation Summary' table is displayed, showing the results for zero-shot, one-shot, and few-shot prompts. The summary indicates that zero-shot prompting identifies sentiment correctly for clear feedback, one-shot prompting improves understanding by providing sentiment reference, and few-shot prompting yields the most accurate results by learning sentiment patterns from multiple examples.

```
[23] ✓ On
Example 2:
Feedback: "The syllabus is outdated."
Sentiment: Negative

Example 3:
Feedback: "Classes were conducted regularly."
Sentiment: Neutral

Now classify the following feedback:
Feedback: "The instructor explained concepts clearly."
===

run_prompt(few_shot_prompt, "Positive")

PROMPT:

Example 1:
Feedback: "Excellent teaching methods."
Sentiment: Positive

Example 2:
Feedback: "The syllabus is outdated."
Sentiment: Negative

Example 3:
Feedback: "Classes were conducted regularly."
Sentiment: Neutral

Now classify the following feedback:
Feedback: "The instructor explained concepts clearly."

MODEL OUTPUT:
Positive
-----

[24] ✓ On
print("Evaluation Summary:")
print("Zero-shot Output : Positive")
print("One-shot Output  : Neutral")
print("Few-shot Output  : Positive")

Evaluation Summary:
Zero-shot Output : Positive
One-shot Output  : Neutral
Few-shot Output  : Positive

Observation

Zero-shot prompting identifies sentiment correctly for clear feedback.
One-shot prompting improves understanding by providing sentiment reference.
Few-shot prompting yields the most accurate results by learning sentiment patterns from multiple examples.
```

Problem Statement 4: Course Recommendation System

The screenshot shows a Jupyter Notebook titled 'Lab_Assignment_4.1_2256.ipynb'. It contains a problem statement and a sample learner queries with corresponding course levels. The code defines a function 'run_prompt' that takes a prompt and an output, and prints the prompt, output, and a separator. It also defines a 'zero_shot_prompt' and a 'few_shot_prompt' for classifying learner queries into Beginner, Intermediate, or Advanced levels. The notebook shows the output of the 'run_prompt' function for the 'zero_shot_prompt' and 'few_shot_prompt'.

```
[27] ✓ On
# Problem statement -4
# Sample learner queries with corresponding course levels

sample_queries = [
    ("I want to learn Python basics.", "Beginner"),
    ("I am new to programming.", "Beginner"),
    ("I know Python and want to learn data structures.", "Intermediate"),
    ("I want to build machine learning models.", "Intermediate"),
    ("I want to master deep learning and transformers.", "Advanced")
]

for query, level in sample_queries:
    print(f"Query: {query}")
    print(f"Level: {level}\n")

... Query: I want to learn Python basics.
Level: Beginner

Query: I am new to programming.
Level: Beginner

Query: I know Python and want to learn data structures.
Level: Intermediate

Query: I want to build machine learning models.
Level: Intermediate

Query: I want to master deep learning and transformers.
Level: Advanced

[28] ✓ On
def run_prompt(prompt, output):
    print("PROMPT:\n")
    print(prompt)
    print("\nMODEL OUTPUT:\n")
    print(output)
    print("-" * 60)

[29] ✓ On
zero_shot_prompt = """
Classify the learner query into:
Beginner, Intermediate, or Advanced.

Query: "I want to learn Python basics."
"""

run_prompt(zero_shot_prompt, "Beginner")

PROMPT:

Classify the learner query into:
Beginner, Intermediate, or Advanced.

Query: "I want to learn Python basics."
```

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands Code Text Run all

Share

RAM Disk

MODEL OUTPUT:

Beginner

[38] one_shot_prompt = """
Example:
Query: "I am new to programming."
Level: Beginner

Now classify the following query:

Query: "I know Python and want to learn data structures."
"""

run_prompt(one_shot_prompt, "Intermediate")

...

PROMPT:

Example:
Query: "I am new to programming."
Level: Beginner

Now classify the following query:

Query: "I know Python and want to learn data structures."

MODEL OUTPUT:
Intermediate

[39] few_shot_prompt = """
Example 1:
Query: "I have no coding experience."
Level: Beginner

Example 2:
Query: "I know Python fundamentals."
Level: Intermediate

Example 3:
Query: "I want to master deep learning models."
Level: Advanced

Now classify the following query:

Query: "I want to learn neural networks from scratch."
"""

run_prompt(few_shot_prompt, "Intermediate")

...

PROMPT:

Example 1:
Query: "I have no coding experience."
Level: Beginner

Example 2:
Query: "I know Python fundamentals."
Level: Intermediate

Example 3:
Query: "I want to master deep learning models."
Level: Advanced

Now classify the following query:

Query: "I want to learn neural networks from scratch."

MODEL OUTPUT:
Intermediate

Variables

Terminal

2:46 PM Python 3

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands Code Text Run all

Share

RAM Disk

[39] Example 2:
Query: "I know Python fundamentals."
Level: Intermediate

Example 3:
Query: "I want to master deep learning models."
Level: Advanced

Now classify the following query:

Query: "I want to learn neural networks from scratch."
"""

run_prompt(few_shot_prompt, "Intermediate")

...

PROMPT:

Example 1:
Query: "I have no coding experience."
Level: Beginner

Example 2:
Query: "I know Python fundamentals."
Level: Intermediate

Example 3:
Query: "I want to master deep learning models."
Level: Advanced

Now classify the following query:

Query: "I want to learn neural networks from scratch."

MODEL OUTPUT:
Intermediate

[40] print("Evaluation Summary:")
print("Zero-shot Output : Beginner")
print("One-shot Output : Intermediate")
print("Few-shot Output : Intermediate")

...

Evaluation Summary:
Zero-shot Output : Beginner
One-shot Output : Intermediate
Few-shot Output : Intermediate

Observation

Zero-shot prompting correctly classifies beginner-level queries.
One-shot prompting improves level detection by giving a reference example.
Few-shot prompting produces the most reliable classification due to multiple skill-level examples.

[] Start coding or generate with AI.

Variables

Terminal

2:46 PM Python 3

Problem Statement 5: Social Media Post Moderation

The first Jupyter Notebook interface displays the initial setup for the social media post moderation task. It includes a file explorer on the left and a top bar with 'Lab_Assignment_4.1_2256.ipynb' and a 'Share' button. The code cells are as follows:

```
[33] ✓ On
# Problem statement - 5
# Sample social media posts with moderation categories

sample_posts = [
    ("Check out our new product launch!", "Acceptable"),
    ("You are useless.", "Offensive"),
    ("Click this link to win a free phone!", "Spam"),
    ("Happy to be part of this community.", "Acceptable"),
    ("Buy now and get 90% discount!", "Spam")
]

for post, category in sample_posts:
    print(f"Post: {post}")
    print(f"Category: {category}\n")
```

The output of the first cell shows the posts and their categories:

```
Post: Check out our new product launch!
Category: Acceptable

Post: You are useless.
Category: Offensive

Post: Click this link to win a free phone!
Category: Spam

Post: Happy to be part of this community.
Category: Acceptable

Post: Buy now and get 90% discount!
Category: Spam
```

```
[34] ✓ On
def run_prompt(prompt, output):
    print("PROMPT:\n")
    print(prompt)
    print("\nMODEL OUTPUT:\n")
    print(output)
    print("-" * 60)
```

```
[35] ✓ On
zero_shot_prompt = """
Classify the following social media post as:
Acceptable, Offensive, or Spam.

Post: "Click this link to win a free phone!"
"""

run_prompt(zero_shot_prompt, "Spam")
```

The output of the second cell shows the prompt and the model's output:

```
PROMPT:

Classify the following social media post as:
Acceptable, Offensive, or Spam.

Post: "Click this link to win a free phone!"

MODEL OUTPUT:

Spam
```

The second Jupyter Notebook interface displays the continuation of the social media post moderation task. It includes a file explorer on the left and a top bar with 'Lab_Assignment_4.1_2256.ipynb' and a 'Share' button. The code cells are as follows:

```
MODEL OUTPUT:

Spam
-----
```

```
[36] ✓ On
one_shot_prompt = """
Example:
Post: "Buy now and get 90% discount!"
Category: Spam

Now classify the following post:

Post: "You are an idiot."
"""

run_prompt(one_shot_prompt, "Offensive")
```

The output of the third cell shows the prompt and the model's output:

```
PROMPT:

Example:
Post: "Buy now and get 90% discount!"
Category: Spam

Now classify the following post:

Post: "You are an idiot."

MODEL OUTPUT:

Offensive
-----
```

```
[37] ✓ On
few_shot_prompt = """
Example 1:
Post: "Check out our new product launch."
Category: Acceptable

Example 2:
Post: "You are useless."
Category: Offensive

Example 3:
Post: "Limited offer! Click now!"
Category: Spam

Now classify the following post:

Post: "Happy to be part of this community."
"""

run_prompt(few_shot_prompt, "Acceptable")
```

The output of the fourth cell shows the prompt and the model's output:

```
PROMPT:

Example 1:
Post: "Check out our new product launch."
Category: Acceptable
```

Lab_Assignment_4.1_2256.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

RAM Disk

[17] Os

Post: "You are useless."
Category: Offensive

Example 3:
Post: "Limited offer! Click now!"
Category: Spam

Now classify the following post:

Post: "Happy to be part of this community."
""

run_prompt(few_shot_prompt, "Acceptable")

... PROMPT:

Example 1:
Post: "Check out our new product launch."
Category: Acceptable

Example 2:
Post: "You are useless."
Category: Offensive

Example 3:
Post: "Limited offer! Click now!"
Category: Spam

Now classify the following post:

Post: "Happy to be part of this community."

MODEL OUTPUT:
Acceptable

[38] Os

print("Evaluation Summary:")
print("Zero-shot Output : Spam")
print("One-shot Output : Offensive")
print("Few-shot Output : Acceptable")

Evaluation Summary:
Zero-shot Output : Spam
One-shot Output : Offensive
Few-shot Output : Acceptable

Observation

Zero-shot prompting works well for obvious spam content but may fail for subtle offensive language.

One-shot prompting improves classification by providing a single reference example.

Few-shot prompting produces the most accurate moderation results by learning from multiple examples.

[] start coding or generate with AI.

Variables Terminal

2:46 PM Python 3