AIAC –[LAB-4.1]

G.RAKESH

2303A51721

Batch-11

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

Week2 -Monday

Problem Statement 0

Lab Objectives:

• To explore and apply different levels of prompt examples in

AI-assisted code generation.

• To understand how zero-shot, one-shot, and few-shot

prompting affect AI output quality.

• To evaluate the impact of context richness and example

quantity on AI performance.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

• Use zero-shot prompting to instruct AI with minimal context.

• Use one-shot prompting with a single example to guide AI code generation.

• Apply few-shot prompting using multiple examples to improve AI responses.

• Compare AI outputs across the three prompting strategies.

Zero-shot Prompting

Prompt Used:

Classify the following news headline into one of these categories:

Politics, Sports, Technology, Entertainment.

Headline: "India wins the T20 cricket series."

Output:

Sports

```
# Classify the following news headline into one of these categories:
# Politics, Sports, Technology, Entertainment.
# Headline: "India wins the T20 cricket series."
# Output:
# Sports give uncommented code with user input and output of the code.
def classify_headline(headline):
    prompt = f"Classify the following news headline into one of these categories: Politics, Sports, Technology, Entertainment.\nHeadline: \"{headline}\""
    return "Sports"
# Example usage
if __name__ == "__main__":
    user_input = input("Enter a news headline: ")
    category = classify_headline(user_input)
    print(f"The headline is classified as: {category}.")
# Explanation of the code.
# The function constructs a prompt to classify the headline and would typically
# call an LLM API to get the classification result.
# The placeholder indicates where the LLM response would be integrated.
```

Headline: "Government announces new education policy"

Category: Politics

Now classify the following headline into Politics, Sports,

Technology, or Entertainment.

Headline: "Tech company launches a new AI-

powered smartphone." Output:

Technology

```
def classify_headline_one_shot(headline):
    example_headline = "Government announces new education policy"
    example_category = "Politics"
    prompt = (f"Example:\nHeadline: \"{example_headline}\"\nCategory: {example_category}\n\n"
              f"Now classify the following headline into Politics, Sports, Technology, or Entertainment.\n"
              f"Headline: \"{headline}\"")
    return "Technology"
# Example usage
if __name__ == "__main__":
    user_input = input("Enter a news headline: ")
    category = classify_headline_one_shot(user_input)
    print(f"The headline is classified as: {category}.")
# Explanation of the code.
# The function includes one example headline and its category in the prompt
# to guide the LLM in classifying the new headline.
# Output:
# The headline is classified as: Technology.
```

Few-shot Prompting

Prompt Used:

Example 1:

Headline: "Parliament passes new tax reform bill"

Category: Politics

Example 2:

Headline: "Football club signs a new international player"

Category: Sports

Example 3:

Headline: "Cybersecurity firm reports major data breach"

Category: Technology

Example 4:

Headline: "Upcoming movie breaks box office records"

Category: Entertainment

Now classify the following headline into Politics, Sports, Technology, or Entertainment.

Headline: "Popular actor announces next film project."

Output:

Entertainment

Observation:

Few-shot prompting produced the most accurate and confident response Problem

Statement1

Customer Email Classification

```python
# generate a python code for the above prompt.
def classify_headline(headline):
    examples = {
        "Parliament passes new tax reform bill": "Politics",
        "Football club signs a new international player": "Sports",
        "Cybersecurity firm reports major data breach": "Technology",
        "Upcoming movie breaks box office records": "Entertainment"
    }

    # Simple keyword-based classification
    politics_keywords = ["parliament", "tax", "government", "election"]
    sports_keywords = ["football", "club", "player", "match"]
    technology_keywords = ["cybersecurity", "data breach", "software", "technology"]
    entertainment_keywords = ["movie", "box office", "actor", "film"]

    headline_lower = headline.lower()

    if any(keyword in headline_lower for keyword in politics_keywords):
        return "Politics"
    elif any(keyword in headline_lower for keyword in sports_keywords):
        return "Sports"
    elif any(keyword in headline_lower for keyword in technology_keywords):
        return "Technology"
    elif any(keyword in headline_lower for keyword in entertainment_keywords):
        return "Entertainment"
    else:
        return "Unknown"
# Example usage
if __name__ == "__main__":
    user_input = input("enter ")
    category = classify_headline(user_input)
    print(f"The headline is classified as: {category}.")
# Explanation of the code.
# The function classify_headline uses keyword-based matching to classify the given headline
# into one of the predefined categories: Politics, Sports, Technology, or Entertainment.
```

Output

```
enter Popular actor announces next film project.
enter Popular actor announces next film project.
The headline is classified as: Entertainment.
```

Problem Statement 1

A company receives a large number of customer emails

every day and wants to automatically classify them into the

following categories:

• Billing

• Technical Support

• Feedback

• Others

Instead of training a new machine learning model, the company

decides to use prompt engineering techniques with an existing large

language model.

Tasks

Task-1

1. Prepare five short sample emails, each belonging to one
   of the above categories.

```python
# generate code that accomplishes the tasks described above in Python for each task
# Task 1: Sample Emails
sample_emails = [
    {
        "category": "Billing",
        "email": "Dear Support, I noticed an unexpected charge on my credit card statement. Can you please help me understand this charge?"
    },
    {
        "category": "Technical Support",
        "email": "Hello, I'm having trouble connecting to the internet on my laptop. It keeps disconnecting every few minutes. Can you assist me with this issue?"
    },
    {
        "category": "Feedback",
        "email": "Hi Team, I just wanted to say that I love the new features in your app! The user interface is very intuitive and easy to navigate."
    },
    {
        "category": "Others",
        "email": "Greetings, I would like to inquire about your company's partnership opportunities. Could you provide more information on how to get started?"
    },
    {
        "category": "Billing",
        "email": "Hello, I need a copy of my last three invoices for my records. Can you send them to me at your earliest convenience?"
    }
]
```

2. Write a zero-shot prompt to classify a given email into
   one of

the categories without providing any examples.

```python
# # Task 2: Zero-Shot Prompt
def classify_email_zero_shot(email):
    prompt = f"Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.\n\nEmail: {email}\nCategory:"
    # Simulated response from a language model
    response = "Billing"  # Placeholder for actual model output
    return response
# Example usage
if __name__ == "__main__":
    test_email = input("Enter : ")
    category = classify_email_zero_shot(test_email)
    print(f"Zero-Shot Classification: {category}")
```

Output:

```
Enter : I have a question about my recent bill. There seems to be an error in the amount charged.
Zero-Shot Classification: Billing
```

3. Write a one-shot prompt by including one labeled email
   example and ask the model to classify a new email.

```python
# # Task 3: One-Shot Prompt
def classify_email_one_shot(email):
    example_email = "Dear Support, I noticed an unexpected charge on my credit card statement. Can you please help me understand this charge?"
    example_category = "Billing"
    prompt = f"Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.\n\nExample Email: {example_email}\nCategory: {exampl
    # Simulated response from a language model
    response = "Billing"  # Placeholder for actual model output
    return response
# Example usage
if __name__ == "__main__":
    test_email = "I'm having trouble connecting to the internet on my laptop. It keeps disconnecting every few minutes."
    category = classify_email_one_shot(test_email)
    print(f"One-Shot Classification: {category}")
```

Output:

```
One-Shot Classification: Billing
```

4. Write a few-shot prompt by including two or three labeled
   email examples and ask the model to classify a new email.

```
# # Task 4: Few-Shot Prompt
def classify_email_few_shot(email):
    example_emails = [
        {
            "email": "Dear Support, I noticed an unexpected charge on my credit card statement. Can you please help me understand this charge?",
            "category": "Billing"
        },
        {
            "email": "Hello, I'm having trouble connecting to the internet on my laptop. It keeps disconnecting every few minutes. Can you assist me with this issue?",
            "category": "Technical Support"
        },
        {
            "email": "Hi Team, I just wanted to say that I love the new features in your app! The user interface is very intuitive and easy to navigate.",
            "category": "Feedback"
        }
    ]
    prompt = "Classify the following email into one of the categories: Billing, Technical Support, Feedback, Others.\n\n"
    for example in example_emails:
        prompt += f"Example Email: {example['email']}\nCategory: {example['category']}\n\n"
    prompt += f"Email: {email}\nCategory:"
    # Simulated response from a language model
    response = "Technical Support"  # Placeholder for actual model output
    return response
# Example usage
if __name__ == "__main__":
    test_email = "I'm having trouble connecting to the internet on my laptop. It keeps disconnecting every few minutes."
    category = classify_email_few_shot(test_email)
    print(f"Few-Shot Classification: {category}")
```

Output:

```
Few-Shot Classification: Technical Support _
```

5. Compare the outputs obtained using zero-shot, one-shot, and few-shot prompting techniques and briefly comment on their effectiveness Problem

Statement 2

Intent Classification for Chatbot Queries

Solution:

The effectiveness of each prompting technique can be summarized as follows:

- Zero-Shot Prompting: This method relies solely on the model's pre-existing

knowledge #   and may not always yield accurate results, especially for specific

or nuanced queries.

- One-Shot Prompting: By providing a single example, this method helps guide the model

towards the desired output, often resulting in improved accuracy compared to zero-shot. # - Few-Shot Prompting: Offering multiple examples allows the model to better understand

the context and nuances of the task, typically leading to the highest accuracy among

the three methods. This approach is particularly effective for complex classification tasks.

Overall, few-shot prompting is generally the most effective technique for intent classification

in chatbot queries, as it provides the model with sufficient context to make informed decisions.

Problem Statement 3

Intent Classification for Chatbot Queries

A company wants to deploy a chatbot to handle customer queries.

Each query must be classified into one of the following

intents: Account Issue, Order Status, Product Inquiry, or

General Question using prompt engineering techniques.

Tasks to be Completed

1. Prepare Sample Data

Create 6 short chatbot user queries, each mapped to one

of the four intents.

2. Zero-shot Prompting

Design a prompt that asks the LLM to classify a user

query into the given intent categories without examples.

3. One-shot Prompting

Provide one labeled query in the prompt before classifying

a new query.

4. Few-shot Prompting

Include 3–5 labeled intent examples to guide the LLM

before classifying a new query.

5. Evaluation

Apply all three techniques to the same set of test queries

and document differences in performance.

```python
# generate uncommented code with user input and output of the code.
def classify_sentiment(feedback):
    positive_keywords = ['good', 'great', 'excellent', 'amazing', 'fantastic', 'love']
    negative_keywords = ['bad', 'poor', 'terrible', 'awful', 'hate', 'worst']

    feedback_lower = feedback.lower()
    if any(word in feedback_lower for word in positive_keywords):
        return "Positive"
    elif any(word in feedback_lower for word in negative_keywords):
        return "Negative"
    else:
        return "Neutral"
if __name__ == "__main__":
    user_input = input("Enter student feedback: ")
    sentiment = classify_sentiment(user_input)
    print(f"The sentiment of the feedback is: {sentiment}.")
# Example usage
# Input:
# Enter student feedback: The course was excellent and the professor was great.
# Output:
# The sentiment of the feedback is: Positive.
# Input:
# Enter student feedback: The course was terrible and I hate the assignments.
# Output:
```

Output

```
Enter student feedback: The course was terrible and I hate the assignments.
The sentiment of the feedback is: Negative.
```

Problem Statement 4

Course Recommendation System

An online learning platform wants to recommend courses by

classifying learner queries into Beginner, Intermediate, or

Advanced

levels.

Questions:

a) Write a Zero-shot prompt to classify learner queries.

b) Create a One-shot prompt with one example query.

c) Develop a Few-shot prompt with multiple labeled queries.

d) Discuss how Few-shot prompting improves
   recommendation

quality.

```python
# generate a python code for the zero-shot prompt.
def classify_query_zero_shot(query):
    beginner_keywords = ['basic', 'introduction', 'getting started', 'fundamentals']
    intermediate_keywords = ['intermediate', 'some experience', 'working knowledge']
    advanced_keywords = ['advanced', 'expert', 'deep dive', 'specialized']

    query_lower = query.lower()

    if any(keyword in query_lower for keyword in beginner_keywords):
        return "Beginner"
    elif any(keyword in query_lower for keyword in intermediate_keywords):
        return "Intermediate"
    elif any(keyword in query_lower for keyword in advanced_keywords):
        return "Advanced"
    else:
        return "Unclassified"
# Example usage
if __name__ == "__main__":
    user_query = input("Enter your learning query: ")
    classification = classify_query_zero_shot(user_query)
    print(f"The query is classified as: {classification}")
# Explanation of the code.
# The function checks for keywords in the user query to classify it into
# Beginner, Intermediate, or Advanced levels based on predefined keyword lists.
# Input:
# Enter your learning query: I want to learn the basics of Python programming.
# Output:
```

Output:

```
Enter your learning query: I want to learn the basics of Python programming.
The query is classified as: Beginner
```

Problem statement -5:

Social Media Post Moderation

A social media platform wants to classify posts into Acceptable,

Offensive, or Spam.

Questions:

a) Write a Zero-shot prompt for post moderation.

b) Convert it into a One-shot prompt.

c) Design a Few-shot prompt using multiple examples.

d) Explain the challenges of Zero-shot prompting in content

   moderation.

```python
# generate a Python function that classifies a social media post
# based on its content.
def classify_post(content):
    offensive_keywords = ["hate", "violence", "abuse"]
    spam_keywords = ["buy now", "click here", "subscribe"]

    content_lower = content.lower()

    if any(keyword in content_lower for keyword in offensive_keywords):
        return "Offensive"
    elif any(keyword in content_lower for keyword in spam_keywords):
        return "Spam"
    else:
        return "Acceptable"
# Example usage
if __name__ == "__main__":
    user_input = input("Enter the social media post content: ")
    classification = classify_post(user_input)
    print(f"The post is classified as: {classification}.")
# Explanation of the code.
# The function checks the content for offensive and spam keywords
# and classifies the post accordingly.
# Zero-shot prompting challenges include lack of context, ambiguity
# in language, and difficulty in handling nuanced content without
```

Output:

```
Enter the social media post content: I hate you
The post is classified as: Offensive.
```