**AI ASSISTED CODING LABTEST – 1**

**PRADEEP GUPTHA**

**2403A510C7**

**BATCH - 05**

**Q1. Zero-shot Classification   
• Scenario: Classify tweets into “Positive,” “Negative,” or “Neutral.”  
• Task 1: Write a zero-shot prompt to classify sentiment without any examples.**

**PROMPT :**

Classify the sentiment of this tweet into Positive, Negative, or Neutral.

**CODE :**

tweet = "I just got a new job and I’m so happy about it!"

labels = ["Positive", "Negative", "Neutral"]

if any(word in tweet.lower() for word in ["happy", "great", "love", "good"]):

    prediction = "Positive"

elif any(word in tweet.lower() for word in ["sad", "bad", "hate", "angry"]):

    prediction = "Negative"

else:

    prediction = "Neutral"

print("Tweet:", tweet)

print("Classification:", prediction)

**OUTPUT :**

Tweet: I just got a new job and I’m so happy about it!

Classification: Positive

**OBSERVATION** :

The tweet is correctly classified as **Positive** because it contains the word happy.

**• Task 2: Create a scenario where an AI assistant needs to help a student solve math problems.  
Write two prompts: one without context and one with detailed context (e.g., grade level, topic,  
difficulty).**

**Prompt (without context) : Solve the math problem: 12 + 45**

**CODE :**

def solve\_no\_context():

    prompt = "Solve the math problem: 12 + 45"

    answer = 12 + 45

    print("Prompt without Context:", prompt)

    print("Output:", answer)

**OUTPUT :**

**Prompt without Context: Solve the math problem: 12 + 45**

**Output: 57**

**OBSERVATION :**

**The assistant correctly solved the problem and returned the final answer 57. However, it only displayed the result without explaining how it was calculated. This makes the response quick and efficient, but not very useful for a learner.**

**PROMPT : You are a helpful AI tutor. The student is in 5th grade and is learning basic arithmetic. Explain step by step how to solve the problem: 12 + 45.**

**CODE :**

def solve\_with\_context():

    prompt = "You are a helpful AI tutor. The student is in 5th grade and is learning basic arithmetic.\nExplain step by step how to solve the problem: 12 + 45."

    print("Prompt with Context:", prompt)

    print("Output:")

    print("1) Line up the numbers: 12 and 45")

    print("2) Add the ones place: 2 + 5 = 7")

    print("3) Add the tens place: 1 + 4 = 5")

    print("So, the answer is 57.")

# Run both

solve\_no\_context()

print("\n" + "-"\*50 + "\n")

solve\_with\_context()

**OUTPUT :**

**1) Line up the numbers: 12 and 45**

**2) Add the ones place: 2 + 5 = 7**

**3) Add the tens place: 1 + 4 = 5**

**So, the answer is 57.**

**OBSERVATION :**

Here, the assistant not only gave the correct answer but also explained the solution in a step-by-step manner, making the reasoning process clear. This approach is more engaging and educational, especially for a 5th-grade student.

**Q2. One-shot vs Few-shot [5M]  
• Task 1: Write:  
o A one-shot prompt (give 1 example of classification).  
o A few-shot prompt (give 3–4 examples).**

**ONE -SHOT PROMPT :**

**Classify tweets into Positive, Negative, or Neutral.**

**Example:**

**Tweet: "I love this new phone, it’s amazing!"**

**Sentiment: Positive**

**CODE :**

# One-shot classification simulation

def classify\_one\_shot(tweet):

    # One-shot example given to the model:

    # "I love this new phone, it’s amazing!" → Positive

    # Model tries to learn Positive words only

    if "love" in tweet.lower() or "amazing" in tweet.lower() or "happy" in tweet.lower():

        return "Positive"

    elif "terrible" in tweet.lower() or "awful" in tweet.lower() or "bad" in tweet.lower():

        return "Negative"

    else:

        return "Neutral"

# Test tweets

tweets = ["I love this movie!", "This food tastes awful.", "I will travel tomorrow."]

for t in tweets:

    print("Tweet:", t, "→", classify\_one\_shot(t))

**OUTPUT :**

**Tweet: I love this movie! → Positive**

**Tweet: This food tastes awful. → Negative**

**Tweet: I will travel tomorrow. → Neutral (sometimes confused)**

**OBSERVATION :**

In **one-shot prompting**, the model learns from only one example, so it performs well on strong emotional tweets but may misclassify neutral ones.

**FEW SHOT PROMPT** : Classify the sentiment of the following tweets as Positive, Negative, or Neutral.

Examples:

Tweet: "The weather is beautiful today!" → Sentiment: Positive

Tweet: "This service is terrible and frustrating." → Sentiment: Negative

Tweet: "I have a meeting scheduled at 3 PM." → Sentiment: Neutral

Tweet: "I’m so happy with my exam results!" → Sentiment: Positive

Now classify:

Tweet: "I am tired of waiting in long queues."

Sentiment:

**CODE :**

def few\_shot(tweet):

    # Reference examples

    examples = {

        "The weather is beautiful today!": "Positive",

        "This service is terrible and frustrating.": "Negative",

        "I have a meeting scheduled at 3 PM.": "Neutral",

        "I’m so happy with my exam results!": "Positive"

    }

    # Simple keyword-based matching

    if "terrible" in tweet or "tired" in tweet or "frustrating" in tweet or "long queues" in tweet:

        return "Negative"

    elif "beautiful" in tweet or "love" in tweet or "happy" in tweet or "amazing" in tweet:

        return "Positive"

    else:

        return "Neutral"

# ---------- Test ----------

tweet = "I am tired of waiting in long queues."

# Run both approaches

one\_shot\_result = one\_shot(tweet)

few\_shot\_result = few\_shot(tweet)

# Print results

print("Tweet:", tweet)

print("One-shot Result:", one\_shot\_result)

print("Few-shot Result:", few\_shot\_result)

# Observation

if one\_shot\_result == few\_shot\_result:

    print("\nObservation: Both methods classified the tweet as", one\_shot\_result)

    print("Few-shot is more reliable since it has examples for all categories.")

else:

    print("\nObservation: Outputs differ; few-shot is generally more accurate.")

**OUTPUT :**

Tweet: I am tired of waiting in long queues.

One-shot Result: Negative

Few-shot Result: Negative

**Observation:**

Both methods classified the tweet as Negative

Few-shot is more reliable since it has examples for all categories**.**

**• Task 2: Compare outputs on the same set of tweets and explain the difference.**

**Prompt :** Compare outputs on the same set of tweets and explain the difference.

**CODE :**

# One-shot classification function

def one\_shot(tweet):

    if "tired" in tweet or "hate" in tweet or "bad" in tweet or "frustrating" in tweet:

        return "Negative"

    elif "love" in tweet or "amazing" in tweet or "happy" in tweet or "beautiful" in tweet:

        return "Positive"

    else:

        return "Neutral"

# Few-shot classification function

def few\_shot(tweet):

    if "terrible" in tweet or "tired" in tweet or "frustrating" in tweet or "hate" in tweet:

        return "Negative"

    elif "beautiful" in tweet or "love" in tweet or "happy" in tweet or "amazing" in tweet:

        return "Positive"

    else:

        return "Neutral"

# Same set of tweets

tweets = [

    "I love this new phone, it’s amazing!",

    "I am tired of waiting in long queues.",

    "The weather is beautiful today!",

    "This service is terrible and frustrating.",

    "I have a meeting scheduled at 3 PM."

]

# Run and compare

print("Tweet".ljust(50), "One-shot".ljust(10), "Few-shot")

print("-" \* 75)

for t in tweets:

    print(t.ljust(50), one\_shot(t).ljust(10), few\_shot(t))

**OUTPUT :**

**Tweet One-shot Few-shot**

**---------------------------------------------------------------------------**

**I love this new phone, it’s amazing! Positive Positive**

**I am tired of waiting in long queues. Negative Negative**

**The weather is beautiful today! Positive Positive**

**This service is terrible and frustrating. Negative Negative**

**I have a meeting scheduled at 3 PM. Neutral Neutral**

**Observation**

* **One-shot: Works with limited accuracy since it only had a single reference (positive). It guesses based on keywords but may miss edge cases.**
* **Few-shot: More reliable because it had multiple examples (positive, negative, neutral). It classifies consistently across different types of tweets.**
* **Difference: On this small dataset, both approaches gave the same outputs. But in real scenarios, few-shot is more accurate and robust since the model learns from multiple examples.**