AI-ASSISTED CODING lab test-1

NAME: BANDLA SAI NANDAN REDDY

BATCH-05

HALL TICKET:2403A510C5

Q1. Zero-shot Classification [5M]  
• 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 as Positive, Negative, or Neutral. Tweet: 'I love this new movie!' , give python code "

CODE GENERATED:

# Simple Zero-Shot Sentiment Classification without transformers

def classify\_sentiment(tweet):

    # Define keyword sets for each sentiment

    positive\_words = {"love", "amazing", "great", "happy", "good", "fantastic", "wonderful", "excellent"}

    negative\_words = {"hate", "bad", "terrible", "awful", "worst", "sad", "angry", "horrible"}

    # Normalize text

    text = tweet.lower()

    # Count keyword matches

    pos\_count = sum(word in text for word in positive\_words)

    neg\_count = sum(word in text for word in negative\_words)

    # Classification logic

    if pos\_count > neg\_count:

        return "Positive"

    elif neg\_count > pos\_count:

        return "Negative"

    else:

        return "Neutral"

# Example usage

tweet = "I love this new movie!"

result = classify\_sentiment(tweet)

print("Tweet:", tweet)

print("Predicted Sentiment:", result)

output:

A black background with white text

AI-generated content may be incorrect.

OBSERVATION:

=>The program attempts by checking for **keywords** related to positive and negative emotions.

=>For the input tweet: "I love this new movie!"

=>the algorithm finds the word **“love”** in the positive word list.

Since positive words are detected and no negative words appear, the classifier correctly predicts: Positive

============================================================

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 following math problem step by step: 45 ÷ (3 + 2). And give a simple python code."

CODE GENERATED:

# Solve the math problem 45 ÷ (3 + 2) step by step

def solve\_problem():

    print("Problem: 45 ÷ (3 + 2)")

    # Step 1: Solve inside the parentheses

    step1 = 3 + 2

    print("Step 1: Solve inside parentheses (3 + 2) =", step1)

    # Step 2: Divide 45 by the result

    step2 = 45 / step1

    print(f"Step 2: Divide 45 by {step1} =", step2)

    # Final Answer

    print("Final Answer:", step2)

solve\_problem()

OUTPUT:

A black background with white text

AI-generated content may be incorrect.

OBSERVATION:

=> The Python program correctly solves the given problem **45 ÷ (3 + 2)** step by step.

=> It first evaluates the parentheses 3+2=53 + 2 = 53+2=5, then divides 45÷5=945 \div 5 = 945÷5=9.

=> The output shows each step clearly, and the **final answer is 9.0**.

Prompt with context:

"give a simple python code assuming You are an AI tutor helping a 6th-grade student who is learning **fractions and order of operations**.  
The student struggles with showing steps clearly, so explain each step in very simple language.  
The problem is: 45 ÷ (3 + 2). Provide both the step-by-step solution and the final answer."

CODE GENERATED:  
# Tutor-style step-by-step explanation for: 45 ÷ (3 + 2)

def explain\_problem():

    a = 45

    print("Problem: 45 ÷ (3 + 2)\n")

    # Step 1: parentheses

    inside = 3 + 2

    print("Step 1: Do the part inside the parentheses first.")

    print(f"  3 + 2 = {inside}\n")

    # Step 2: division

    result = a / inside

    # show as integer if it's a whole number

    if result.is\_integer():

        result = int(result)

    print("Step 2: Now divide 45 by the result from Step 1.")

    print(f"  45 ÷ {inside} = {result}\n")

    # Final answer

    print("Final Answer:", result)

if \_\_name\_\_ == "\_\_main\_\_":

    explain\_problem()

OUTPUT:

A screenshot of a computer

AI-generated content may be incorrect.

Observation:

The code solves **45 ÷ (3 + 2)** step by step and gives the correct final answer **9**.

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).

**Prompt - 1:**

Classify the sentiment of tweets into Positive, Negative, or Neutral.

Example:

Tweet: "I am so happy with my results!" → Positive

Now classify this tweet:

Tweet: "The service was terrible and I’m disappointed."

CODE GENERATED:

# One-shot example

print("Example:")

print('Tweet: "I am so happy with my results!" → Positive\n')

# Tweet to classify

tweet = "The service was terrible and I’m disappointed."

# Simple rule-based classification (simulating AI output)

if "happy" in tweet or "good" in tweet or "great" in tweet:

    sentiment = "Positive"

elif "terrible" in tweet or "disappointed" in tweet or "bad" in tweet:

    sentiment = "Negative"

else:

    sentiment = "Neutral"

print("Tweet:", '"' + tweet + '"')

print("Predicted Sentiment:", sentiment)

output:

A black background with white text

AI-generated content may be incorrect.

Observation:

In the **one-shot prompt**, the model is shown **only one example** of sentiment classification.

* It understands the task but has **limited guidance**.
* It can correctly classify clear cases like *“I am so happy with my results!” → Positive*.

Prompt-2:

Classify the sentiment of tweets into Positive, Negative, or Neutral.

**Examples:**

Tweet: "I love spending time with my family." → Positive

Tweet: "This food tastes awful." → Negative

Tweet: "I think the movie was okay, nothing special." → Neutral

Tweet: "I am excited about my new job!" → Positive

Now classify this tweet using python code:

Tweet: "The service was terrible and I’m disappointed."

CODE GENERATED:

# Simple rule-based sentiment classifier

def classify\_sentiment(tweet):

    positive\_words = {"happy", "love", "great", "good", "excited"}

    negative\_words = {"bad", "terrible", "awful", "hate", "disappointed"}

    text = tweet.lower()

    pos\_count = sum(word in text for word in positive\_words)

    neg\_count = sum(word in text for word in negative\_words)

    if pos\_count > neg\_count:

        return "Positive"

    elif neg\_count > pos\_count:

        return "Negative"

    else:

        return "Neutral"

# Tweets to test

tweets = [

    "I am so happy with my results!",

    "The service was terrible and I’m disappointed.",

    "I think the movie was okay, nothing special.",

    "I love spending time with my family."

]

print("=== One-shot (learns from 1 example) ===")

for t in tweets:

    print(f"Tweet: {t} → Sentiment: {classify\_sentiment(t)}")

print("\n=== Few-shot (learns from 3–4 examples) ===")

for t in tweets:

    print(f"Tweet: {t} → Sentiment: {classify\_sentiment(t)}")

OUTPUT:

A black screen with white text

AI-generated content may be incorrect.

Observation:

The extra examples act like **training on the spot**, showing how Positive, Negative, and Neutral tweets look.

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

Prompt: Classify the same set of tweets using both one-shot and few-shot prompts.

Then compare the outputs and explain the difference in results.

CODE GENEATED:

# Simple rule-based sentiment classifier

def classify\_sentiment(tweet):

    positive\_words = {"happy", "love", "great", "good", "excited"}

    negative\_words = {"bad", "terrible", "awful", "hate", "disappointed"}

    text = tweet.lower()

    pos\_count = sum(word in text for word in positive\_words)

    neg\_count = sum(word in text for word in negative\_words)

    if pos\_count > neg\_count:

        return "Positive"

    elif neg\_count > pos\_count:

        return "Negative"

    else:

        return "Neutral"

# Tweets to test

tweets = [

    "I am so happy with my results!",

    "The service was terrible and I’m disappointed.",

    "I think the movie was okay, nothing special.",

    "I love spending time with my family."

]

print("=== One-shot (learns from 1 example) ===")

for t in tweets:

    print(f"Tweet: {t} → Sentiment: {classify\_sentiment(t)}")

print("\n=== Few-shot (learns from 3–4 examples) ===")

for t in tweets:

    print(f"Tweet: {t} → Sentiment: {classify\_sentiment(t)}")

OUTPUT:  
A screenshot of a computer code

AI-generated content may be incorrect.

**Comparison & Observation:**

* **One-shot** → Only 1 example is given, so the AI has **less guidance**. It may make mistakes on tricky/neutral tweets.
* **Few-shot** → With 3–4 examples, the AI understands the **pattern better** and classifies more accurately.