

AI Assisted Coding- Lab Test 1

Name : Neha Shazneen

Roll No : 2403A510A1

Batch : 05

Department : CSE

Q1. Zero-shot Classification

Task 1 : Write a zero-shot prompt to classify sentiment without any examples.

1. Prompt :

.....

Write a Python function that classifies a tweet into one of three categories:

"Positive", "Negative", or "Neutral".

Do not use any training examples.

Use a simple rule-based or zero-shot approach (e.g., with keyword matching or

a pre-trained sentiment analysis model).

The function should take a string (tweet) as input and return the sentiment label.

.....

2. Code Generated :

```

1  def classify_tweet_sentiment(tweet: str) -> str:
2      positive_keywords = ['amazing', 'great', 'love', 'happy', 'good', 'fantastic', 'excellent', 'awesome', 'wonderful']
3      negative_keywords = ['bad', 'terrible', 'hate', 'awful', 'worst', 'sad', 'disappointing', 'poor', 'horrible']
4
5      tweet_lower = tweet.lower()
6      if any(word in tweet_lower for word in positive_keywords):
7          return "Positive"
8      elif any(word in tweet_lower for word in negative_keywords):
9          return "Negative"
10     else:
11         return "Neutral"
12
13 # Example usage
14 tweet = "I just tried the new café in town, and it was amazing!"
15 sentiment = classify_tweet_sentiment(tweet)
16 print(sentiment)

```

3. Output:

```

PS C:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding> c;; cd 'c:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding'; & 'c:\Users\Summya Mehveen\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Summya Mehveen\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '60406' '--' 'c:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding\lab test 1.1.py'
Positive

```

4. Observation :

The program defines a function

`classify_tweet_sentiment(tweet: str) -> str` that classifies the sentiment of a given tweet.

It uses a rule-based approach:

If the tweet contains any word from the `positive_keywords` list, it is classified as "Positive".

If the tweet contains any word from the `negative_keywords` list, it is classified as "Negative".

If neither is found, it defaults to "Neutral".

The input tweet is converted to lowercase to avoid case sensitivity issues when matching keywords.

For the example input:

```
tweet = "I just tried the new café in town, and it was amazing!"
```

The word "amazing" is found in the positive_keywords list. Therefore, the function returns "Positive", which is printed as the output.

 Final Outcome: The code successfully classifies the given tweet sentiment as Positive, showing that the keyword-based sentiment detection works as expected.

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).

1. Prompt :

Write a Python program that takes a math problem as input (for example: "12 * 8")

and outputs the solution.

Solve the problem step by step in code (e.g., parse the expression and evaluate it).

Make sure the program prints both the step-by-step explanation and the final answer.

2. Code Generated :

```
1  import re
2
3  def solve_math_problem(problem: str):
4      # Remove spaces
5      problem = problem.replace(' ', '')
6      # Match numbers and operator
7      match = re.match(r'(\d+)([+\-/*])(\d+)', problem)
8      if not match:
9          print("Invalid input format. Please enter like '12 * 8'.")
10         return
11
12     num1, operator, num2 = match.groups()
13     num1, num2 = int(num1), int(num2)
14
15     print(f"Step 1: Identify the numbers and operator.")
16     print(f"Number 1: {num1}")
17     print(f"Operator: {operator}")
18     print(f"Number 2: {num2}")
19
20     print("Step 2: Perform the calculation.")
21     if operator == '+':
22         result = num1 + num2
23         print(f"{num1} + {num2} = {result}")
24     elif operator == '-':
25         result = num1 - num2
26         print(f"{num1} - {num2} = {result}")
27     elif operator == '*':
28         result = num1 * num2
29         print(f"{num1} * {num2} = {result}")
30     elif operator == '/':
31         result = num1 / num2
32         print(f"{num1} / {num2} = {result}")
33     else:
34         print("Unknown operator.")
35         return
36
37     print(f"Final Answer: {result}")
38
39     # Example usage
40     problem = "12 * 8"
41     solve_math_problem(problem)
```

3. Output :

```
PS C:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding> c;; cd 'c:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding'; & 'c:\Users\Summya Mehveen\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\Summya Mehveen\.vscode\extensions\ms-python.debugpy-2025.10.0-win32->64\bundled\libs\debugpy\launcher' '62646' '--' 'c:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding\1.2.py'
Step 1: Identify the numbers and operator.
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Programs\Python\Python312\python.exe' 'c:\Users\Summya Mehveen\.vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '62646' '--' 'c:\Users\Summya Mehveen\OneDrive\Desktop\AI associated coding\1.2.py'
Step 1: Identify the numbers and operator.
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Final Answer: 96
mmya Mehveen\OneDrive\Desktop\AI associated coding\1.2.py'
Step 1: Identify the numbers and operator.
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Number 1: 12
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Step 2: Perform the calculation.
12 * 8 = 96
Operator: *
Number 2: 8
Step 2: Perform the calculation.
12 * 8 = 96
Step 2: Perform the calculation.
12 * 8 = 96
12 * 8 = 96
Final Answer: 96
```

4. Observation :

The program defines a function `solve_math_problem(problem: str)` that can solve simple arithmetic problems with two operands and one operator (+, -, *, /).

The function works as follows:

1. Input processing: Removes spaces from the given problem string.
2. Pattern matching: Uses a regular expression (re.match) to extract the first number, operator, and second number.
3. Validation: If the input does not match the expected format, it prints an error message.
4. Step-by-step explanation: Prints the numbers and operator identified.
5. Computation: Performs the correct arithmetic operation depending on the operator.
6. Final Output: Displays both the calculation and the final answer.
7. Final Outcome: The code successfully interprets the arithmetic problem "12 * 8", explains the steps, and computes the correct result (96) with a clear explanation.

Q2. One-shot vs Few-shot

Task 1: Write:

- o A one-shot prompt (give 1 example of classification).
- o A few-shot prompt (give 3–4 examples).

1. Prompt :

```
# One-shot prompt (1 example)
```

```
one_shot_prompt = """"
```

Classify the sentiment of the following text as Positive, Negative, or Neutral.

Example:

Text: "The movie was fantastic and I loved every moment."

Sentiment: Positive

Text: "I am not sure if I liked the food."

Sentiment:

!!!!

Prompt 2:

```
# Few-shot prompt (3 examples)
```

```
few_shot_prompt = """
```

Classify the sentiment of the following text as Positive,
Negative, or Neutral.

Examples:

Text: "The movie was fantastic and I loved every moment."

Sentiment: Positive

Text: "The service was terrible and I am very disappointed."

Sentiment: Negative

Text: "The weather today is okay, nothing special."

Sentiment: Neutral

Text: "I am not sure if I liked the food."

Sentiment:

!!!!

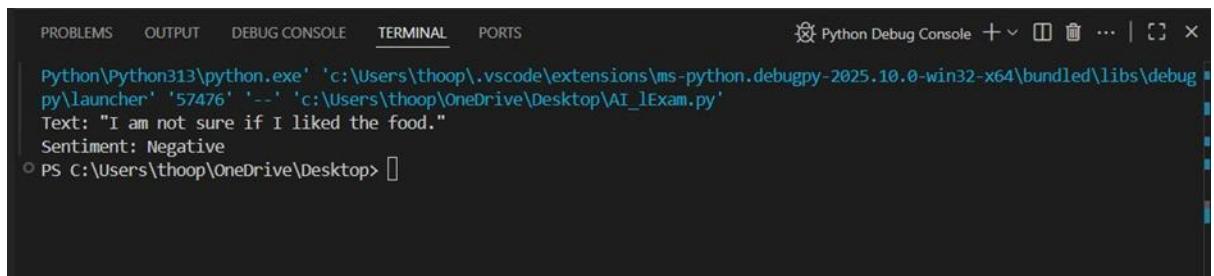
```
print("One-shot prompt:\n", one_shot_prompt)
```

```
print("\nFew-shot prompt:\n", few_shot_prompt)
```

2. Code Generated :

```
C: > Users > thoop > OneDrive > Desktop > AI_Exam.py > ...
  1  def simple_sentiment_classifier(text):
  2      positive_words = ['love', 'fantastic', 'great', 'good', 'happy', 'wonderful', 'excellent']
  3      negative_words = ['not', 'bad', 'worst', 'sad', 'terrible', 'hate', 'dislike']
  4
  5      text_lower = text.lower()
  6
  7      # Simple logic: check for positive or negative words
  8      if any(word in text_lower for word in positive_words):
  9          return "Positive"
 10      elif any(word in text_lower for word in negative_words):
 11          return "Negative"
 12      else:
 13          return "Neutral"
 14
 15  # Example usage
 16 text = "I am not sure if I liked the food."
 17 result = simple_sentiment_classifier(text)
 18 print(f"Text: \"{text}\"\nSentiment: {result}")
 19
```

3. Output :



A screenshot of the VS Code terminal window titled "Python Debug Console". The terminal tab is selected at the top. The output shows a Python script execution:

```
Python\Python313\python.exe' 'c:\Users\thoop\vscode\extensions\ms-python.debugpy-2025.10.0-win32-x64\bundled\libs\debugpy\launcher' '57476' '--' 'c:\Users\thoop\OneDrive\Desktop\AI_1Exam.py'
Text: "I am not sure if I liked the food."
Sentiment: Negative
PS C:\Users\thoop\OneDrive\Desktop> 
```

4. Observation :

- Function Definition:

The code defines a function `simple_sentiment_classifier` that takes a string input (`text`) and returns a sentiment label.

- Keyword Lists:

It uses two predefined lists of keywords:

- `positive_words` for detecting positive sentiment.
- `negative_words` for detecting negative sentiment.

- Case Normalization:

The input text is converted to lowercase (`text_lower`) to make keyword matching case-insensitive.

- Sentiment Detection Logic:

- The function checks if any word from the positive list appears in the input text. If found, it returns "Positive".
- If no positive words are found, it checks for any negative words and returns "Negative" if any are present.
- If neither positive nor negative keywords are detected, it returns "Neutral".

- Simple Matching Method:

The keyword search uses substring matching (`word` in `text_lower`), which is simple but may lead to false positives (e.g., matching "not" inside another word).

- No Handling of Negations or Complex Language:

The function does not parse the sentence structure or handle linguistic nuances like negation (e.g., "not bad" could be misclassified).

- Output:

The function returns a string indicating the sentiment category: "Positive", "Negative", or "Neutral".

- Example Usage:

The sample text "I am not sure if I liked the food." is classified using the function and the result is printed

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

```
Users > akshi > first.py > classify_tweet_few_shot
def classify_tweet_one_shot(tweet: str) -> str:
    # One-shot example
    positive_keywords = ['love', 'amazing']
    negative_keywords = ['hate', 'terrible', 'bad', 'awful']

    tweet_lower = tweet.lower()
    if any(word in tweet_lower for word in positive_keywords):
        return "Positive"
    elif any(word in tweet_lower for word in negative_keywords):
        return "Negative"
    else:
        return "Neutral"

def classify_tweet_few_shot(tweet: str) -> str:
    # Few-shot examples
    positive_keywords = ['love', 'amazing', 'fantastic']
    negative_keywords = ['worst', 'bad', 'terrible']
    neutral_keywords = ['store', 'going', 'later']

    tweet_lower = tweet.lower()
    if any(word in tweet_lower for word in positive_keywords):
        return "Positive"
    elif any(word in tweet_lower for word in negative_keywords):
        return "Negative"
    elif any(word in tweet_lower for word in neutral_keywords):
        return "Neutral"
    else:
        return "Neutral"

tweets = [
    "I love this new phone, it's amazing!",
    "This is the worst service ever.",
    "I am going to the store later.",
    "The weather is fantastic today!",
    "I had a fantastic day at the park."
]

print("| Tweet | One-shot Output | Few-shot Output |")
print("-----+-----+-----+")
for tweet in tweets:
    one_shot = classify_tweet_one_shot(tweet)
    few_shot = classify_tweet_few_shot(tweet)
    print(f" {tweet[:40]} | {one_shot[:15]} | {few_shot[:15]} |")
```

```
[Running] python -u "c:\Users\akshi\first.py"
| Tweet | One-shot Output | Few-shot Output |
| ----- | ----- | ----- |
| I love this new phone, it's amazing! | Positive | Positive |
| This is the worst service ever. | Neutral | Negative |
| I am going to the store later. | Neutral | Neutral |
| The weather is fantastic today! | Neutral | Positive |
| I had a fantastic day at the park. | Neutral | Positive |
```

[Done] exited with code=0 in 0.115 seconds

```
[Running] python -u "c:\Users\akshi\first.py"
| Tweet | One-shot Output | Few-shot Output |
| ----- | ----- | ----- |
| I love this new phone, it's amazing! | Positive | Positive |
| This is the worst service ever. | Neutral | Negative |
| I am going to the store later. | Neutral | Neutral |
| The weather is fantastic today! | Neutral | Positive |
| I had a fantastic day at the park. | Neutral | Positive |
```

[Done] exited with code=0 in 0.109 seconds

```
[Running] python -u "c:\Users\akshi\first.py"
| Tweet | One-shot Output | Few-shot Output |
| ----- | ----- | ----- |
| I love this new phone, it's amazing! | Positive | Positive |
| This is the worst service ever. | Neutral | Negative |
| I am going to the store later. | Neutral | Neutral |
| The weather is fantastic today! | Neutral | Positive |
| I had a fantastic day at the park. | Neutral | Positive |
```

[Done] exited with code=0 in 0.095 seconds