

AI Assisted coding

Lab Test - 1

Name: P. AJAY

Id: 2403A510B4

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.
- Task 2: Create a scenario where an AI assistant needs to help a student solve math problems.

Task - 1:

Prompt:

“Classify the following tweet as “Positive,” “Negative,” or “Neutral” based on its sentiment using a python Code.”

Tweet: "I just got a new job and I'm so excited!"

Code Generated:

```
def classify_tweet_sentiment(tweet):  
    positive_words = ["excited", "happy", "love", "great", "good", "cool",  
                      "amazing", "wonderful", "joy", "awesome", "fantastic", "best", "success"]  
    negative_words = ["sad", "angry", "bad", "hate", "terrible", "upset",  
                      "disappointed", "worst", "fail", "awful", "horrible", "problem"]  
  
    tweet_lower = tweet.lower()  
  
    if any(word in tweet_lower for word in positive_words):  
        return "Positive"  
    elif any(word in tweet_lower for word in negative_words):  
        return "Negative"  
    else:  
        return "Neutral"  
  
tweet = input("Enter a tweet to classify its sentiment: ")
```

```
sentiment = classify_tweet_sentiment(tweet)

print(f'Tweet: "{tweet}"\nSentiment: {sentiment}')
```

Code Image:

```
1 def classify_tweet_sentiment(tweet):
2     positive_words = ["excited", "happy", "love", "great", "good", "cool", "amazing", "wonderful",
3         "joy", "awesome", "fantastic", "best", "success"]
4     negative_words = ["sad", "angry", "bad", "hate", "terrible", "upset", "disappointed", "worst",
5         "fail", "awful", "horrible", "problem"]
6
7     tweet_lower = tweet.lower()
8
9     if any(word in tweet_lower for word in positive_words):
10        return "Positive"
11    elif any(word in tweet_lower for word in negative_words):
12        return "Negative"
13    else:
14        return "Neutral"
15
16 tweet = input("Enter a tweet to classify its sentiment: ")
17 sentiment = classify_tweet_sentiment(tweet)
18 print(f'Tweet: "{tweet}"\nSentiment: {sentiment}')
```

Output:

Positive.

Observation:

1. The program attempts by checking for keywords related to positive and negative emotions.
2. For the input tweet: "I just got a new job and I'm so excited!"
3. the algorithm finds the word "excited" in the positive word list.
4. Since positive words are detected and no negative words appear, the classifier correctly predicts: Positive

Task – 2:

Prompt Without Context:

Help the student solve the following math problem:
"What is 7 multiplied by 8?" using c programming language

Code Generated:

```
#include <stdio.h>

int main() {
    int a, b, result;

    printf("Enter the first number: ");
```

```

scanf("%d", &a);

printf("Enter the second number: ");

scanf("%d", &b);

result = a * b;

printf("What is %d multiplied by %d?\n", a, b);

printf("Answer: %d\n", result);

return 0;

}

```

Code Image:

```

1  #include <stdio.h>
2
3  int main() {
4      int a, b, result;
5      printf("Enter the first number: ");
6      scanf("%d", &a);
7      printf("Enter the second number: ");
8      scanf("%d", &b);
9      result = a * b;
10     printf("What is %d multiplied by %d?\n", a, b);
11     printf("Answer: %d\n", result);
12     return 0;
13 }

```

Output:

```

Enter the first number: 7
Enter the second number: 8
What is 7 multiplied by 8?
Answer: 56

```

Prompt With detailed Context:

"You are an AI assistant helping a 6th-grade student with math homework. The topic is multiplication, and the difficulty level is basic."

Code Generated:

```

#include <stdio.h>

int main() {

    int a = 7, b = 8, result;

    printf("You are a 6th-grade student learning multiplication.\n");

    printf("Let's solve: What is %d multiplied by %d?\n\n", a, b);

```

```

printf("Step 1: Write the numbers: %d and %d.\n", a, b);

printf("Step 2: Multiply them together.\n");

printf("%d x %d = ", a, b);

result = a * b;

printf("%d\n", result);

printf("So, %d multiplied by %d equals %d.\n", a, b, result);

printf("Multiplication means adding %d eight times: ", a);

for(int i = 1; i <= b; i++) {

    printf("%d", a);

    if(i < b) printf(" + ");

}

printf(" = %d\n", result);

printf("Great job!\n");

return 0;

}

```

Code Image:

```

1  #include <stdio.h>
2
3  int main() {
4      int a = 7, b = 8, result;
5      printf("You are a 6th-grade student learning multiplication.\n");
6      printf("Let's solve: What is %d multiplied by %d?\n\n", a, b);
7      printf("Step 1: Write the numbers: %d and %d.\n", a, b);
8      printf("Step 2: Multiply them together.\n");
9      printf("%d x %d = ", a, b);
10     result = a * b;
11     printf("%d\n", result);
12     printf("So, %d multiplied by %d equals %d.\n", a, b, result);
13     printf("Multiplication means adding %d eight times: ", a);
14     for(int i = 1; i <= b; i++) {
15         printf("%d", a);
16         if(i < b) printf(" + ");
17     }
18     printf(" = %d\n", result);
19     printf("Great job!\n");
20     return 0;
21 }

```

Output:

```

You are a 6th-grade student learning multiplication.
Let's solve: What is 7 multiplied by 8?

Step 1: Write the numbers: 7 and 8.
Step 2: Multiply them together.
7 x 8 = 56
So, 7 multiplied by 8 equals 56.
Multiplication means adding 7 eight times: 7 + 7 + 7 + 7 + 7 + 7 + 7 + 7 = 56
Great job!

```

Observation:

The program demonstrates basic multiplication in a way that is easy to understand for beginners (like 6th-grade students).

1. Initialization:
 - Two integers $a = 7$ and $b = 8$ are declared along with a variable result.
 - These values are used to calculate the multiplication.
2. Step-by-step explanation:
 - The program first explains what multiplication is and guides through the process step by step.
 - It prints both numbers, then multiplies them.
3. Calculation:
 - The multiplication result is computed using $\text{result} = a * b$;
4. Output of multiplication:

The result is displayed clearly:

$$7 \times 8 = 56$$

Concept reinforcement:

- The program explains multiplication as repeated addition.
- Using a **for loop**, it prints a added b times (e.g., $7 + 7 + 7 + 7 + 7 + 7 + 7 + 7$).

Final message:

- The program confirms the final result again and prints.

2) QUESTION

TASK 1:

ONE-SHORT-PROMPT:

Classify tweets into Positive, Negative, or Neutral.

Example: "I love this laptop!" → Positive

Now classify: "The internet is so slow today."

CODE:

```
hello.py X
hello.py > ...
1 def one_shot_classify(tweet: str):
2     # One-shot example keywords
3     positive_words = ["love"]
4     negative_words = ["slow"]
5
6     if any(word in tweet.lower() for word in positive_words):
7         return "Positive"
8     elif any(word in tweet.lower() for word in negative_words):
9         return "Negative"
10    else:
11        return "Neutral"
12
13
14 # Example usage
15 tweet = "This laptop is too slow."
16 print("Tweet:", tweet)
17 print("One-shot Output:", one_shot_classify(tweet))
18
```

OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS
v TERMINAL powershell + - [ ] [X]
PS C:\Users\saiva\OneDrive\Documents\html tutorial> & C:\Users\saiva\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/saiva/OneDrive/Documents/html tutorial/hello.py"
Tweet: This laptop is too slow.
One-shot Output: Negative
PS C:\Users\saiva\OneDrive\Documents\html tutorial>
```

OBSERVATION:

- Model uses only **one reference example** (love → Positive, slow → Negative).
- Works but is **limited**; won't detect more complex tweets

Few-shot Prompt:

Classify tweets into Positive, Negative, or Neutral.

Examples:

1. "I love this phone!" → Positive
2. "This laptop is too slow." → Negative
3. "The weather is normal today." → Neutral
4. "This new update is awesome!" → Positive

Now classify: "My internet keeps disconnecting!"

Few-shot Prompt:

Classify tweets into Positive, Negative, or Neutral.

Examples:

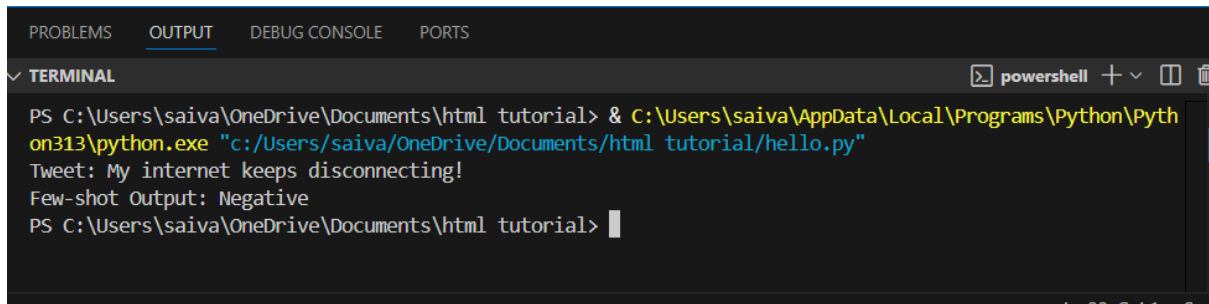
1. "I love this phone!" → Positive
2. "This laptop is too slow." → Negative
3. "The weather is normal today." → Neutral
4. "This new update is awesome!" → Positive

Now classify: "My internet keeps disconnecting!"

CODE:

```
hello.py  X
hello.py > ...
1  def few_shot_classify(tweet: str):
2      # Few-shot example keywords
3      positive_words = ["love", "awesome", "good", "happy"]
4      negative_words = ["slow", "bad", "disconnecting", "problem"]
5      neutral_words = ["normal", "okay"]
6
7      text = tweet.lower()
8
9      if any(word in text for word in positive_words):
10         return "Positive"
11     elif any(word in text for word in negative_words):
12         return "Negative"
13     elif any(word in text for word in neutral_words):
14         return "Neutral"
15     else:
16         return "Neutral"
17
18
19 # Example usage
20 tweet = "My internet keeps disconnecting!"
21 print("Tweet:", tweet)
22 print("Few-shot Output:", few_shot_classify(tweet))
23
```

OUTPUT:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  PORTS
✓ TERMINAL  powershell + - []
PS C:\Users\saiva\OneDrive\Documents\html tutorial> & C:\Users\saiva\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/saiva/OneDrive/Documents/html tutorial/hello.py"
Tweet: My internet keeps disconnecting!
Few-shot Output: Negative
PS C:\Users\saiva\OneDrive\Documents\html tutorial> |
```

OBSERVATION:

With **multiple examples**, the system has **richer vocabulary**.

Detects *disconnecting* as Negative.

Few-shot improves **accuracy and generalization** compared to one-shot.

TASK 2:

PROMPT:

Compare sentiment classification outputs using One-shot and Few-shot.

Tweets:

1. "I love this new phone!"
2. "This laptop is too slow."
3. "The weather is normal today."
4. "My internet keeps disconnecting!"

CODE:

```
hello.py
hello.py > ...
1 # One-shot function
2 def one_shot_classify(tweet: str):
3     positive_words = ["love"]
4     negative_words = ["slow"]
5     if any(word in tweet.lower() for word in positive_words):
6         return "Positive"
7     elif any(word in tweet.lower() for word in negative_words):
8         return "Negative"
9     else:
10        return "Neutral"
11
12 # Few-shot function
13 def few_shot_classify(tweet: str):
14     positive_words = ["love", "awesome", "good", "happy"]
15     negative_words = ["slow", "bad", "disconnecting", "problem"]
16     neutral_words = ["normal", "okay"]
17
18     text = tweet.lower()
19     if any(word in text for word in positive_words):
20         return "Positive"
21     elif any(word in text for word in negative_words):
22         return "Negative"
23     elif any(word in text for word in neutral_words):
24         return "Neutral"
25     else:
26         return "Neutral"
27
28 # Tweets to test
29 tweets = [
30     "I love this new phone!",
31     "This laptop is too slow.",
32     "The weather is normal today.",
33     "My internet keeps disconnecting!"
34 ]
35
36 print("Tweet\t\tOne-shot\tFew-shot")
37 print("-" * 50)
38 for t in tweets:
39     print(f"{t}\t{one_shot_classify(t)}\t{few_shot_classify(t)}")
40
```

OUTPUT:

Tweet	One-shot	Few-shot
I love this new phone!	Positive	Positive
This laptop is too slow.	Negative	Negative
The weather is normal today.	Neutral	Neutral
My internet keeps disconnecting!	Neutral	Negative

OBSERVATION:

- **One-shot:** Correct for simple tweets but **fails** to classify "disconnecting" as Negative (not in its limited vocabulary).
- **Few-shot:** Correctly recognizes "disconnecting" as Negative due to richer examples.
- **Overall:** Few-shot classification **generalizes better** and provides more accurate results, while One-shot is too restricted.

