AI Assistant Coding

Assignment-4.1

Name:K.Trinay Prasad

HT No:2303A52070

Batch:32

Q1. Zero-Shot Prompting (Basic Lab Task)

Task: Write a Python function that classifies a given text as Spam or Not Spam

using zero-shot prompting.

Steps:

1. Construct a prompt without any examples.

2. Clearly specify the output labels.

3. Display only the predicted label.

Prompt:

```
#Write a python function that classifies a text  as spam or not
```

Code:

```python
def classify_text(text):
    spam_keywords = ["free", "win", "winner", "cash", "prize", "urgent", "click here"]
    text_lower = text.lower()
    for keyword in spam_keywords:
        if keyword in text_lower:
            return "spam"
    return "not spam"
#Example usage
if __name__ == "__main__":
    test_text=["Congratulations! You are a winner of a cash prize.",
               "Hello, how are you doing today?",
               "Click here to claim your free gift now!",
               "Don't forget our meeting tomorrow."]
    for text in test_text:
        if classify_text(text)=="spam":
            print(f"'{text}' is classified as spam.")
        else:
            print(f"'{text}' is not classified as spam.")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
'Congratulations! You are a winner of a cash prize.' is classified as spam.
'Hello, how are you doing today?' is not classified as spam.
'Click here to claim your free gift now!' is classified as spam.
'Don't forget our meeting tomorrow.' is not classified as spam.
PS D:\AI Assicoding>
```

Q2. One-Shot Prompting (Emotion detection)

Task: Write a Python program that detects the emotion of a sentence using

one-shot prompting.

Emotions: [‘happy', 'sad', 'angry', 'excited', 'nervous', ’neutral’]

Steps:

1. Provide one labeled example inside the prompt.

2. Take a sentence as input.

3. Print the predicted emotion.

Prompt:

```
'''Write a python function that prints emotions of a senetence
Emotions: ['happy', 'sad', 'angry', 'excited', 'nervous', 'neutral']'''
```

Code:

```
def classify_emotion(text):
    emotions_keywords = {
        "happy": ["joy", "pleased", "delighted", "content"],
        "sad": ["unhappy", "sorrowful", "dejected", "downcast"],
        "angry": ["mad", "furious", "irate", "annoyed"],
        "excited": ["thrilled", "elated", "overjoyed", "eager"],
        "nervous": ["anxious", "worried", "tense", "apprehensive"],
        "neutral": ['calm', 'indifferent', 'unemotional', 'detached','impartial']
    }
    text_lower = text.lower()
    for emotion, keywords in emotions_keywords.items():
        for keyword in keywords:
            if keyword in text_lower:
                return emotion #classified emotion
    return "neutral"#Default to neutral if no keywords found
#Example usage
if __name__ == "__main__":
    test_text=[
        "I am so happy and delighted today!",
        "She felt very anxious before the exam.",
        "He was furious about the delay.",
        "They are thrilled about the upcoming trip.",
        "It's just an ordinary day."
    ]
    for text in test_text:
        emotion = classify_emotion(text)
        print(f"Emotion:{emotion}| Text:'{text}'")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Emotion:happy| Text:'I am so happy and delighted today!'
Emotion:nervous| Text:'She felt very anxious before the exam.'
Emotion:angry| Text:'He was furious about the delay.'
Emotion:excited| Text:'They are thrilled about the upcoming trip.'
Emotion:neutral| Text:'It's just an ordinary day.'
PS D:\AI Assicoding>
```

Q3. Few-Shot Prompting (Student Grading Based on Marks)

Task: Write a Python program that predicts a student's grade based on marks

using few-shot prompting.

Grades:

['A', 'B', 'C', 'D', 'F']

Grading Criteria (to be inferred from examples):

- **90–100 → A**
- **80–89 → B**
- **70–79 → C**
- **60–69 → D**
- **Below 60 → F**

Prompt:

```
'''Write a python program to grade students based on their marks
example:
90-100: A
80-89: B
70-79: C
60-69: D
Below 60: F'''
```

Code:

```
def grade_student(marks):
    if 90 <= marks <= 100:
        return 'A'
    elif 80 <= marks < 90:
        return 'B'
    elif 70 <= marks < 80:
        return 'C'
    elif 60 <= marks < 70:
        return 'D'
    elif 0 <= marks < 60:
        return 'F'
    else:
        return 'Invalid marks'  # Handle invalid input
#Example usage
if __name__ == "__main__":
    test_marks = [95, 82, 76, 64,  50, -10, 105]
    for marks in test_marks:
        grade = grade_student(marks)
        print(f"Marks: {marks} | Grade: {grade}")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Marks: 95 | Grade: A
Marks: 82 | Grade: B
Marks: 76 | Grade: C
Marks: 64 | Grade: D
Marks: 50 | Grade: F
Marks: -10 | Grade: Invalid marks
Marks: 105 | Grade: Invalid marks
PS D:\AI Assicoding>
```

Q4. Multi-Shot Prompting (Indian Zodiac Sign Prediction using

Month Name)

Task: Write a Python program that predicts a person's Indian Zodiac sign (Rashi)

based on the month of birth (month name) using multi-shot prompting. Indian

Zodiac Order (Simplified Month-Based Model): The Indian Zodiac cycle starts in

March with Mesha and follows this order:

March → Mesha

April → Vrishabha

May → Mithuna

June → Karka

July → Simha

August → Kanya

September → Tula

October → Vrischika

November → Dhanu

December → Makara

January → Kumbha

February → Meena

Prompt:

```
'''Write a python program that predicts a person zodiac sign based on
birthmonth
signs in order:
March → Mesha
April → Vrishabha
May → Mithuna
June → Karka
July → Simha
August → Kanya
September → Tula
October → Vrischika
November → Dhanu
December → Makara
January → Kumbha
February → Meena '''
```

Code:

```python
def predict_zodiac_sign(month):
    zodiac_signs = {
        "march": "Mesha",
        "april": "Vrishabha",
        "may": "Mithuna",
        "june": "Karka",
        "july": "Simha",
        "august": "Kanya",
        "september": "Tula",
        "october": "Vrischika",
        "november": "Dhanu",
        "december": "Makara",
        "january": "Kumbha",
        "february": "Meena"
    }
    month_lower = month.lower()
    return zodiac_signs.get(month_lower, "Invalid month")  # Handle invalid input
#Example usage
if __name__ == "__main__":
    test_months = ["March", "July", "November", "January", "InvalidMonth"]
    for month in test_months:
        sign = predict_zodiac_sign(month)
        print(f"Birth Month: {month} | Zodiac Sign: {sign}")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Birth Month: March | Zodiac Sign: Mesha
Birth Month: July | Zodiac Sign: Simha
Birth Month: November | Zodiac Sign: Dhanu
Birth Month: January | Zodiac Sign: Kumbha
Birth Month: InvalidMonth | Zodiac Sign: Invalid month
PS D:\AI Assicoding>
```

Q5. Result Analysis Based on Marks

Task: Write a Python program that determines whether a student Passes or

Fails based on marks using Chain-of-Thought (CoT)

prompting.

Result Categories:

['Pass', 'Fail']

Prompt:

```
'''Write a python program to determine if a student has passed or failed
based on their marks
the program should take the marks as input and print "Pass" if the marks
are 40 or above,and "Fail" if the marks are below 40.'''
```

Code:

```
marks=float(input("Enter the marks obtained by the student: "))
if marks>=40:
    print("Pass")
else:
    print("Fail")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Enter the marks obtained by the student: 90
Pass
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Enter the marks obtained by the student: 30
Fail
PS D:\AI Assicoding>
```

Q6 Voting Eligibility Check (Chain-of-Thought Prompting)

Task: Write a Python program that determines whether a person is eligible to

vote using Chain-of-Thought (CoT) prompting.

Prompt:

```
'''Write a well commented python program to determine if a person is eligible to vote or not
based on their age.
The program should take the age as input and print "Eligible to vote" if the age is 18 or above,
and "Not eligible to vote" if the age is below 18.
check for invalid input (negative age) and handle it appropriately.'''
```

Code:

```
try:
    age = int(input("Enter your age: "))#Taking user input and converting it to an integer
    if age < 0:
        print("Invalid input: Age cannot be negative.")#Handling negative age input
    elif age >= 18:
        print("Eligible to vote")#If age is 18 or above, the person is eligible to vote
    else:
        print("Not eligible to vote")#If age is below 18, the person is not eligible to vote
except ValueError:
    print("Invalid input: Please enter a valid integer for age.")#Handling non-integer input
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Enter your age: 25
Eligible to vote
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Not eligible to vote
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Enter your age: -18
Invalid input: Age cannot be negative.
PS D:\AI Assicoding>
```

Q7 Prompt Chaining (String Processing – Palindrome Names)

Task: Write a Python program that uses the prompt chaining technique to

identify palindrome names from a list of student names

Prompt:

```
'''1.generate a list of student names
2.check if the given name is palindrome or not
3.display the list of palindromic names'''
```

Code:

```
def is_palindrome(name):
    return name.lower() == name.lower()[::-1]

student_names=["Anna","Bob","Cathy","David","Eve","Hannah",
               "John","Level","Mike","Noon"]
palindromic_names=[]
for name in student_names:
    if is_palindrome(name):
        palindromic_names.append(name)
print("Palindromic names are:",palindromic_names)
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
Palindromic names are: ['Anna', 'Bob', 'Eve', 'Hannah', 'Level', 'Noon']
PS D:\AI Assicoding>
```

Q8 Prompt Chaining (String Processing – Word Length Analysis)

Task: Write a Python program that uses prompt chaining to analyze a list of

words. In the first prompt, generate a list of words. In the second prompt,

traverse the list and calculate the length of each word. In the third prompt, use

the output of the previous step to determine whether each word is Short

(length less than 5) or Long (length greater than or equal to 5), and display the

result for each word

Prompt:

```
'''1.Generate a list of random words
2.traverse words in the list and find the length
3.from above result check the word is short if less than 5 and long if more than 5 or equal to 5
4.display the result as output'''
```

Code:

```
def classify_word_length(word):
    length=len(word)
    if length<5:
        return "short"
    elif length>5:
        return "long"
    else:
        return "equal"
words=["apple","banana","cherry","date","elderberry","fig","grape","honeydew","kiwi","lemon "]
for word in words:
    length_category=classify_word_length(word)
    print(f"The word {word} is {length_category} (length: {len(word)})")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Ass4.1.py"
The word apple is equal (length: 5)
The word banana is long (length: 6)
The word cherry is long (length: 6)
The word date is short (length: 4)
The word elderberry is long (length: 10)
The word fig is short (length: 3)
The word grape is equal (length: 5)
The word honeydew is long (length: 8)
The word kiwi is short (length: 4)
The word lemon  is long (length: 6)
PS D:\AI Assicoding>
```
