ASSIGNMENT-4
VELPULA VIGNESH
23O3A51707
BATCH-24

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.
Input:
"Congratulations! You have won a free lottery ticket."
Expected Output:
Spam

INPUT:

assn.py > ...

```python
#generate a well commented python function that classifies a given text as spam or not.
def classify_text(text):
    # Define keywords commonly found in spam messages
    spam_keywords = [
        "urgent", "limited time", "act now", "click here", "free money",
        "no obligation", "risk free", "guarantee", "congratulations",
        "winner", "prize money", "cash bonus","Congratulations! You have won a free lottery ticket."
    ]

    # Convert text to lowercase for case-insensitive matching
    text_lower = text.lower()

    # Count occurrences of spam keywords
    spam_count = sum(1 for keyword in spam_keywords if keyword in text_lower)

    # Classify as spam if more than 2 keywords are found
    if spam_count >= 1:
        return True  # Spam
    else:
        return False  # Not Spam
# Example usage
user_input = input("Enter the text to classify: ")
if classify_text(user_input):
    print("The text is classified as SPAM.")
else:
    print("The text is classified as NOT SPAM.")
print("Task 1 completed.")
```

OUTPUT:

```
Enter the text to classify: Congratulations! You have won a free lottery ticket.
The text is classified as SPAM.
Task 1 completed.
```

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

INPUT:

```python
#Generate a well commented python program that detects the emotion of a sentence.
'''input:I am so happy today!
   output:Happy
   input:I am very sad right now.
   output:Sad
   input:I am angry about the delay.
   output:Angry
   input:I feel scared in the dark.
   output:Scared
   input:I am surprised by the news.
   output:Surprised
   '''
def detect_emotion(sentence):
    # Define keywords associated with different emotions
    emotion_keywords = {
        "happy": ["happy", "joy", "pleased", "delighted", "excited", "glad"],
        "sad": ["sad", "unhappy", "sorrowful", "dejected", "downcast"],
        "angry": ["angry", "mad", "furious", "irate", "annoyed"],
        "scared": ["scared", "afraid", "fearful", "terrified", "nervous"],
        "surprised": ["surprised", "astonished", "amazed", "startled"]
    }

    # Convert sentence to lowercase for case-insensitive matching
    sentence_lower = sentence.lower()

    # Check for keywords in the sentence and determine the emotion
    for emotion, keywords in emotion_keywords.items():
        if any(keyword in sentence_lower for keyword in keywords):
            return emotion.capitalize()

    return "Emotion not detected"
# Example usage
user_sentence = input("Enter a sentence to detect emotion: ")
detected_emotion = detect_emotion(user_sentence)
print(f"The detected emotion is: {detected_emotion}")
print("Task 2 completed.")
```

OUTPUT:

```
Enter a sentence to detect emotion: i am very happy
The detected emotion is: Happy
Task 2 completed.
```

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

INPUT:

```
#generate a well commented python program that predicts a student's grade based on their marks and try exception handling.
'''
input:95
output:A
input:85
output:B
input:75
output:C
input:65
output:D
input:55
output:F
'''
def predict_grade(marks):
    """Predict the grade based on the marks obtained."""
    try:
        marks = float(marks)
        if marks < 0 or marks > 100:
            return "Marks should be between 0 and 100."

        if marks >= 90:
            return "A"
        elif marks >= 80:
            return "B"
        elif marks >= 70:
            return "C"
        elif marks >= 60:
            return "D"
        else:
            return "F"
    except ValueError:
        return "Invalid input. Please enter numeric marks."
# Example usage
user_marks = input("Enter the student's marks: ")
predicted_grade = predict_grade(user_marks)
print(f"The predicted grade is: {predicted_grade}")
print("Task 3 completed.")
```

OUTPUT:

```
Enter the student's marks: 95
The predicted grade is: A
Task 3 completed.

Task 3 completed.
```

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

INPUT:

```python
#generate a well commented python program that predicts a person's indian zodiac sign based on their birth year.
'''input:march
   output:mesha
   input:april
   output:vrsabha
   input:may
   output:mithuna
   input:june
   output:karka
   input:july
   output:simha
   input:august
   output:kanya
   input:september
   output:tula
   input:october
   output:vrscika
   input:november
   output:dhanus
   input:december
   output:makara
   input:january
   output:kumbha
   input:february
   output:meena
   '''
def indian_zodiac_sign(month):
    """Return the Indian zodiac sign based on the birth month."""
    zodiac_signs = {
        "march": "Mesha",
        "april": "Vrishabha",
        "may": "Mithuna",
        "june": "Karka",
        "july": "Simha",
        "august": "Kanya",
        "september": "Tula",
        "october": "Vrscika",
        "november": "Dhanus",
        "december": "Makara",
        "january": "Kumbha",
        "february": "Meena"
    }

    month_lower = month.lower()
    return zodiac_signs.get(month_lower, "Invalid month. Please enter a valid month name.")
# Example usage
user_month = input("Enter your birth month: ")
zodiac_sign = indian_zodiac_sign(user_month)
print(f"Your Indian zodiac sign is: {zodiac_sign}")
print("Task 4 completed.")
```

OUTPUT:

```
Enter your birth month: august
Your Indian zodiac sign is: Kanya
Task 4 completed.
```

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']

INPUT:

```
'''
Generate a well commemted python code that determines whether a student passes or fails based on marks
it should handle invalid inputs using exception handling.
it should store in a list
'''
def determine_pass_fail(marks_list):
    """Determine whether each student in the marks list passes or fails."""
    results = []
    for marks in marks_list:
        try:
            marks = float(marks)
            if marks < 0 or marks > 100:
                results.append("Invalid marks. Please enter marks between 0 and 100.")
            elif marks >= 40:
                results.append("Pass")
            else:
                results.append("Fail")
        except ValueError:
            results.append("Invalid input. Please enter numeric marks.")
    return results
# Example usage
user_marks_input = input("Enter the marks of students separated by commas: ")
marks_list = user_marks_input.split(',')
pass_fail_results = determine_pass_fail(marks_list)
for i, result in enumerate(pass_fail_results):
    print(f"Student {i+1}: {result}")
print("Task 5 completed.")
```

OUTPUT:

```
Task 4 completed.
Enter the marks of students separated by commas: 45,23,88,99,
Student 1: Pass
Student 2: Fail
Student 3: Pass
Student 4: Pass
Student 5: Invalid input. Please enter numeric marks.
Task 5 completed.
```

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.

INPUT:

```
'''
Generate a python code  wether s person is eligible to vote or not
'''
def is_eligible_to_vote(age):
    """Check if a person is eligible to vote based on their age."""
    try:
        age = int(age)
        if age < 0:
            return "Invalid age. Age cannot be negative."
        elif age >= 18:
            return "Eligible to vote."
        else:
            return "Not eligible to vote."
    except ValueError:
        return "Invalid input. Please enter a numeric age."

# Example usage
user_age = input("Enter your age: ")
eligibility = is_eligible_to_vote(user_age)
print(eligibility)
print("Task 6 completed.")
```

OUTPUT:

```
Enter your age: 19
Eligible to vote.
Task 6 completed.
```

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.

INPUT:

```
'''
Generate a python code that uses the prompt chaining
technique to identify palindrome names from a list of student
names.
'''
def is_palindrome(name):
    """Check if a given name is a palindrome."""
    name_cleaned = name.replace(" ", "").lower()  # Remove spaces and convert to lowercase
    return name_cleaned == name_cleaned[::-1]
# Example usage
user_names_input = input("Enter student names separated by commas: ")
names_list = [name.strip() for name in user_names_input.split(',')]
palindrome_names = [name for name in names_list if is_palindrome(name)]
print("Palindrome names in the list are:")
for name in palindrome_names:
    print(name)
print("Task 7 completed.")
```

OUTPUT:

```
Enter student names separated by commas: vinnu,abhi
Palindrome names in the list are:
Task 7 completed.
```

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

INPUT:

```
'''
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
'''
def analyze_word_lengths(words):
    """Analyze the length of each word and classify as Short or Long."""
    results = {}
    for word in words:
        length = len(word)
        classification = "Short" if length < 5 else "Long"
        results[word] = (length, classification)
    return results
# Example usage
user_words_input = input("Enter words separated by commas: ")
words_list = [word.strip() for word in user_words_input.split(',')]
word_length_analysis = analyze_word_lengths(words_list)
print("Word Length Analysis:")
for word, (length, classification) in word_length_analysis.items():
    print(f"Word: '{word}', Length: {length}, Classification: {classification}")
print("Task 8 completed.")
```

OUTPUT:

```
Enter words separated by commas: apple,ball,cat
Word Length Analysis:
Word: 'apple', Length: 5, Classification: Long
Word: 'ball', Length: 4, Classification: Short
Word: 'cat', Length: 3, Classification: Short
Task 8 completed.
```