Topic 1: Prompt Engineering

Assignment 1: Zero-shot vs Few-shot Prompting

Objective

The objective of this assignment is to understand the difference between **zero-shot** and **few-shot prompting** in Large Language Models (LLMs). Zero-shot prompting means giving the model a direct query without examples, while few-shot prompting provides a few labeled examples before the query to guide the model's response.

Input:

```python
from transformers import pipeline

classifier = pipeline("text-classification", model="distilbert-base-uncased-finetuned-sst-2-english")

sentence = "I love my new phone."
zero_shot_prompt = "Determine if the following sentence is positive or negative: " + sentence
zero_shot_output = classifier(sentence)

few_shot_prompt = """Example 1: I am happy today. → Positive
Example 2: I am sad today. → Negative
Now classify: The movie was amazing."""
few_shot_sentence = "The movie was amazing."
few_shot_output = classifier(few_shot_sentence)

print("Zero-shot prompt:", zero_shot_prompt)
print("Zero-shot output:", zero_shot_output)
print("Few-shot prompt:\n", few_shot_prompt)
print("Few-shot output:", few_shot_output)
```

Output:

```
Device set to use cpu
Zero-shot prompt: Determine if the following sentence is positive or negative: I love my new phone.
Zero-shot output: [{'label': 'POSITIVE', 'score': 0.999670147895813}]
Few-shot prompt:
 Example 1: I am happy today. → Positive
Example 2: I am sad today. → Negative
Now classify: The movie was amazing.
Few-shot output: [{'label': 'POSITIVE', 'score': 0.9998829364776611}]
```

Observations:

For **simple tasks like sentiment analysis**, both zero-shot and few-shot prompting gave correct results.

Zero-shot works well for straightforward queries but may struggle with ambiguous sentences.

Few-shot prompting provides more context and improves accuracy, especially for more complex or domain-specific tasks.

In this experiment, both approaches gave the same output ("Positive"), but few-shot prompting would likely outperform in more nuanced cases.

Assignment 2: Role-based & Chain-of-Thought Prompting

- **Role-based prompting** tailors tone and complexity to the audience (e.g., high school students).
- **Step-by-step prompting** ensures structured reasoning, making complex processes easier to follow systematically.

Input:

Assignment 2: Role-based & Chain-of-Thought Prompting

```python
!pip -q install transformers accelerate sentencepiece

from transformers import pipeline

gen = pipeline("text2text-generation", model="google/flan-t5-base")

role_prompt = "You are a high school biology teacher. Explain photosynthesis to students in simple words."
step_prompt = "Explain photosynthesis step by step, reasoning each step clearly."

out_role = gen(role_prompt, max_length=220, do_sample=False)[0]["generated_text"]
out_step = gen(step_prompt, max_length=260, do_sample=False)[0]["generated_text"]

print("=== Role-Based Output ===\n", out_role, "\n")
print("=== Step-by-Step Output ===\n", out_step)
```

Output:

```
config.json: 0.00B [00:00, ?B/s]
model.safetensors:   0%|          | 0.00/990M [00:00<?, ?B/s]
generation_config.json:   0%|          | 0.00/147 [00:00<?, ?B/s]
tokenizer_config.json: 0.00B [00:00, ?B/s]
spiece.model:   0%|          | 0.00/792k [00:00<?, ?B/s]
tokenizer.json: 0.00B [00:00, ?B/s]
special_tokens_map.json: 0.00B [00:00, ?B/s]
Device set to use cpu
Both `max_new_tokens` (=256) and `max_length`(=220) seem to have been set. `max_new_tokens` will take precedence. Please refer to the d
Both `max_new_tokens` (=256) and `max_length`(=260) seem to have been set. `max_new_tokens` will take precedence. Please refer to the d
=== Role-Based Output ===
 Photosynthesis is the process of converting sunlight into light energy by photosynthesis.

=== Step-by-Step Output ===
 Photosynthesis is the process of converting sunlight into light energy. Light energy is converted into light energy by photosynthesis.
```

Both the Tasks are performed in Google colab link:

https://colab.research.google.com/drive/1oeuy1utGoKXHvFbcWci3Rybqslv4hmbQ?usp=sharing