Assignment 2: Chain-of-Thought for Python Coding

Due: 2024-10-10 15:29 PM

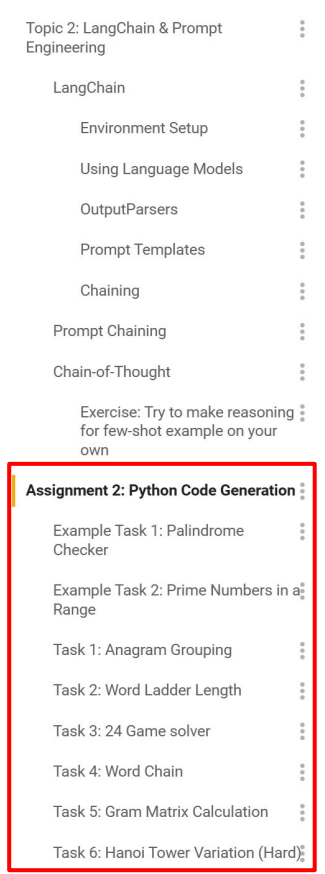
Tasks

1. Apply Chain-of-Thought for 4 tasks: **5 pts**
2. Analysis: **5 pts**

Total 10 pts

Python Notebook for Assignment 2

- Use the same Python Notebook file with the TA session.
  - “**PE2024F_Assignment_2.ipynb**” file from eTL
- Will use lower part of the notebook for the assignment 2



1. Apply Chain-of-Thought for the Tasks

- For given **6 tasks** and descriptions, you should **select 4 tasks** and **create CoT** to guide the LLM in generating Python code.
- The Python code **must contain test code** with at least **three examples** to verify that the code works correctly.

Example Task 2: Prime Numbers in a Range

Create a Python function `find_primes(a, b)` that returns a list of all prime numbers between two integers `a` and `b` (inclusive).

```
[44] prompt_template = PromptTemplate(
         input_variables=["problem_statement", "cot"],
         template="""
         Generate a python code for the following problem statement:

         {problem_statement}

         {cot}
         """
     )

     chain = prompt_template | model | parser


     problem_statement = """
     A Python function find_primes(a, b) that returns a list of all prime numbers between two integers a and b (inclusive).
     """

     ################ TODO: Modify here ################

     cot = """
     First, you should define a function is_prime() to check whether the given number is a prime number or not.
     And then, define a find_primes() function that finds all prime numbers between given two integers(inclusive).
     After defining functions, you should write a test code with at least three examples.
     """

     ###################################################

     result = chain.invoke({"problem_statement": problem_statement, "cot": cot})

     print(result)
```

1. Apply Chain-of-Thought for the Tasks

- You should **run and test the code** by copying and pasting the LLM output.
- Please include the screenshots of the generated Python code and the output of it.
- You will get **1.25 points for each task**

```python
################ TODO: Paste the code here ################
def is_prime(n):
    """Check if a number is prime."""
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def find_primes(a, b):
    """Return a list of all prime numbers between two integers a and b (inclusive)."""
    primes = []
    for num in range(a, b + 1):
        if is_prime(num):
            primes.append(num)
    return primes

# Test code
if __name__ == "__main__":
    # Test case 1: Primes between 10 and 30
    print("Primes between 10 and 30:", find_primes(10, 30))

    # Test case 2: Primes between 1 and 20
    print("Primes between 1 and 20:", find_primes(1, 20))

    # Test case 3: Primes between 50 and 60
    print("Primes between 50 and 60:", find_primes(50, 60))
```

Python code generated by LLM

2. Analysis

- The analysis part must include an analysis of the following:
  - How you designed Chain-of-Thought to solve the task.
  - How the CoT improves each code.
  -
- There is no amount limit for the analysis section, so using LLMs for generating analysis is not effective. So just analyze on your own, do not use LLM for generating analysis report.

Submission

- Due: 10/10 (Thu), 15:29 PM (right before class)
- After you have completed the assignment, you need to submit your report as a PDF via eTL assignment submission
  - Ex) 2024-00000.pdf
- We will calculate deducted score according to the following equation:
  - (Original_score) × (1 − 0.1 × ceil(delayed_days))
  - After 5 days (120 hours), there is no score.
  - Ex1) 50 hours late: 70% of the original score
  - Ex2) 115 hours late: 50% of the original score
  - Ex3) 120 hours late: 0% of the original score
- Concept level discussion is encouraged, but discussion of code/prompt directly related to assignments is not allowed. The assignments must be students' own work.