Assignment 5: Agent

Due: 2024-12-03 15:29 PM

Tasks

1. Improve LLM Agent for Wordle Solver
2. Run Test Set and Get Result: **10 pts**

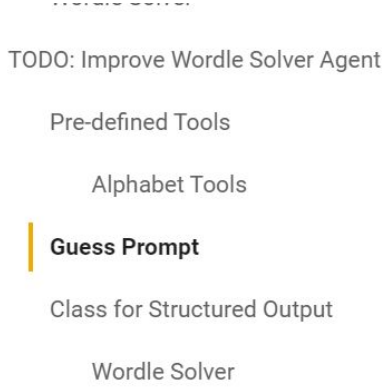
Total 10 pts

Python Notebook for Assignment 5

- Download the Python Notebook from the eTL.
  - “**PE2024F_Assignment_5.ipynb**” file
- Or you can download the Notebook file from the Google Drive
  - https://colab.research.google.com/drive/1iPpV3dKpRezn32gKLr4ZiQL-KIonUwR_?usp=drive_link

1. Improve LLM Agent for Wordle Solver

- For given skeleton codes, you should improve the Wordle solver by **using tools, modifying guess prompt**...
- You can also modify the other part of code freely to fit your agent design.



Guess Prompt

```
[ ]  from langchain_core.prompts import PromptTemplate
     guess_prompt = PromptTemplate(
         input_variables= ####### TODO #######

         template="""
         ####### TODO #######
         """
     )
```

Class for Structured Output

```
[ ]  from typing import List
     from pydantic import BaseModel, Field

     # TypedDict
     class
     ####### TODO #######
```

2. Run Test Set and Get Result:

- **Run the test code** at the below part of the Notebook.
- Your score will be calculated as below:
  - score = {solve_ratio} * 20
- You will get **10 points** if the solve_ratio is larger than 0.5
- If your solve_ratio is the highest, then you will get a chance for presentation
  - If there are students with same solve_ratio, then compare the average solve tries

```
test_word_list_20 = ['Crisp', 'Blown', 'Heart', 'Gloom', 'Flare', 'Juicy', 'Vapid', 'Mirth', 'Pluck', 'Grave',
                     'Snipe', 'Batch', 'Drown', 'Lofty', 'Quake', 'Shark', 'Moped', 'Whirl', 'Grasp', 'Fjord']
test_word_list_10 = ["Trend", "Brisk", "Giant", "Mouth", "Flick", "Spear", "Dwell", "Grimy", "Vouch", "Knead"]
```

```
def test_wordle_solver(test_word_list):
  """Tests the Wordle solver with a list of words and calculates success rate and average solve tries.

  Args:
    test_word_list: A list of words to test the solver with.

  Returns:
    A tuple containing the success rate and average solve tries.
  """
  success_count = 0
  total_tries = 0

  for word in test_word_list:
    print("Given word: " + word)
    solution, tries = solve_wordle(word)
    if solution:
      success_count += 1
      total_tries += tries
    print("--------------------------")

  if success_count > 0:
    success_rate = (success_count / len(test_word_list)) * 100
    average_tries = total_tries / success_count
  else:
    success_rate = 0
    average_tries = 0

  return success_rate, average_tries

# Test the Wordle solver with the test_word_list
success_rate, average_tries = test_wordle_solver(test_word_list_20)
# success_rate, average_tries = test_wordle_solver(test_word_list_10)
```

Test code - it takes few minutes

Submission

- Due: 12/03 (Tue), 15:29 PM (right before class)
- After you have completed the assignment, you need to submit your **Python Notebook File or Python file as a .ipynb or .py** via eTL assignment submission
  - Ex) 2024-00000.ipynb or 2024-00000.py
  - If you cannot upload the file, you can also upload compressed .zip file.
- 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.