
Attention-Deficient Agreeable Agent¹

Kevin Vegda
kevinvegda@gmail.com

With
Cooperative AI & Apart Research

Abstract

A common scenario during game theoretic situations is to check on how an agreeable agent would react to games like a Prisoner's Dilemma or a Stag Hunt. Our agent has one simple trait in terms of its architecture - it is agreeable. Moreover, we give it a periodic reminder to stay on topic to keep it from veering too far from the topic of discussion.

Keywords: Cooperative Agents, Concordia, DeepMind, Agreeableness

1. Introduction

Simulating social scenarios has been a common application that has been tried out in different ways after the emergence of Large Language Models². Running different scenarios with an LLM backend can help potentially predict how two real-life agents would interact. (though the level of correlation is to be established)

During the Concordia hackathon, we tried different approaches to modeling these agents to see which ones among them would be robust enough to deal with different scenarios that require cooperation - from haggling to labor strikes.

2. Overview

Our agent has one defining trait - it is agreeable. It tries to agree with what other agents suggest and thus tries to minimize conflict. This might not give the optimal

¹ Research conducted at the [Concordia Contest Hackathon](#) 2024

outcome in all scenarios, especially sophisticated ones, but it helps to see how agents with different models of mind operate in these game theoretic scenarios.

Moreover, to deal with smaller models like codestral wandering from the topic of discussion, we added a reminder component - named after the pomodoro method - to keep reminding the agent to be on topic.

3. Code

Since the starting point was the `basic_agent` on the `concordia` repo, we only reproduce the code for the components we added on top:

...

```
def get_pomodoro_reminder(overarching_goal, agent_name):

    def pomodoro_reminder(chain_of_thought, clock_now):

        """Reminds the agent to stay on topic"""

        return f'{agent_name} should stay on topic: {overarching_goal}.'

    return pomodoro_reminder

pomodoro_reminder = get_pomodoro_reminder(config.goal, agent_name)

pomodoro_label = (

    f'\nRemind {agent_name} to stay locked in on the task at hand'

)

pomodoro = agent_components.scheduled_hint.ScheduledHint(

    model=model,

    components={

        _get_class_name(observation_summary): observation_summary_label,

        _get_class_name(situation_perception):

situation_perception_label,

        _get_class_name(time_display): 'The current date/time is',
```

```

    },

    clock_now=clock.now,

    hints=[pomodoro_reminder],

    pre_act_key=pomodoro_label,

    logging_channel=measurements.get_channel('PomodoroHint').on_next,

)

```

```

agreeableness_label = f'\nMake {agent_name} agreeable to everything'

agreeableness = agent_components.constant.Constant(

    state=f'{agent_name} agrees.',

    pre_act_key=agreeableness_label,

    logging_channel=measurements.get_channel('Agreeableness').on_next,

)

```

...

4. Discussion and Conclusion

As we can see from the evaluation scores available, making the basic agent more agreeable and giving it a reminder to stay on topic does change the scores a little. Further refining the agent to make the reminder conditional on whether the agent is staying on track based on a goal and some sort of similarity score is a potential direction to pursue.