
Very Cooperative Agent¹

Jakub Fidler

With
Cooperative AI & Apart Research

Abstract

This project explores the development of cooperative AI agents within the Concordia framework, aiming to address challenges in creating agents capable of effective cooperation in various scenarios. The key issue identified was the frequent failure of the Codestral LLM, which was used in the final evaluation, to understand the situation, such as confusing available options and agent roles. The main insight from this limitation was that complex reasoning was unreliable with this LLM, so a strategy that consistently prioritized cooperation was adopted.

The agent designed for this contest always attempted to cooperate, which proved effective, especially in "resident" scenarios where multiple copies of the agent were present. This strategy used the fact that most agents cooperating lead to a high score for these agents. However, in "visitor" scenarios, where only one agent interacted with predefined agents, the strategy's performance was more mixed. To address difficulties of the agent to identify a cooperative action in some scenarios, such as the pub coordination scenario, conditional instructions were developed to improve the agent's reliability.

Performance improved through iterative adjustments, but the project revealed broader issues, including the limitations of predefined agents and the need for agents to understand their real options and communication limits. The project concludes that future cooperative AI should incorporate more advanced reasoning, defecting strategies, and an understanding of game theory, potentially enhancing cooperative interactions in diverse and unpredictable environments.

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

Keywords: Cooperative Agents

1. Introduction

The field of Cooperative AI is concerned with problems of cooperation and how they relate to AI. One of the goals of Cooperative AI is to build AI agents which are good at cooperating, be it with other AI agents, or humans.

The Concordia framework is a “library to facilitate construction and use of generative agent-based models to simulate interactions of agents in grounded physical, social, or digital space”.²

I participated in a contest in which AI agents were evaluated on a range of scenarios from the Concordia framework. The goal of the participants was to get a high score on the evaluation. The purpose of the contest was to make progress on creating cooperative AI agents and to improve the effectiveness of the Concordia framework as a testing ground for these agents.

2. Overview

After some tests with the Concordia framework using the Codestral LLM (which was used during the final evaluation), I noticed multiple things.

The Codestral model was often making basic mistakes. For example, it said that 4 coins per kilogram is a lower price than 3 coins per kilogram. (“The average market price for fruit is 3 coins per kilogram, but Juniper is offering a lower price of 4 coins per kilogram.”)

Furthermore, it also often got confused with the roles of other agents, for example whether it was selling or buying in the haggling scenario.

Formative memories and details about the scenarios were distracting agents from the decisions that mattered. For example, the rational agent (provided as an example in the Concordia framework repository), has a component which considers options that are available to the agent, and then tries to choose the best one. The issue is that the agent was considering choices which were not available to it. For example, it was considering going to dinner with the other workers in the labor collective agent scenario. This is not a choice that agents can make in the scenario. (The only choices are whether to join the strike or go to work.)

Due to these observations, I decided to use a strategy which doesn’t depend on the model being competent at complex reasoning.

² <https://github.com/google-deepmind/concordia>

I analyzed the provided scenarios to see which actions generally lead to high scores. There are “resident” scenarios and “visitor” scenarios. In resident scenarios, there are multiple copies of the agent to be evaluated. In visitor scenarios, there is only one copy of the agent to be evaluated. In resident scenarios it is mostly good for the agent to be cooperative, because its copies will be cooperative as well, leading to a high score for the group (even if some non-evaluated agent defects). In visitor scenarios, in general, defecting leads to a high score if the other agents cooperate, and to a low score otherwise.

It’s important to note, that the submitted agents only interacted with a couple of predefined agents during the evaluation. (They did not interact with agents submitted by other participants.) These predefined agents were not defecting often.

In the end, I decided to make an agent which always tries to cooperate. The reasoning behind that was that it should work reasonably well in most scenarios. It would probably be very good in the resident scenarios. Also, there would be no agents which aggressively take advantage of the cooperative behavior of my agent, as the predefined agents didn’t do that, at least from what I observed.

My intention was to make an agent which always cooperates, however cooperating is not always as easy as choosing the “cooperate” action. For instance, in the pub coordination scenario, each agent decides which pub they personally go to. In general, more agents deciding on the same pub leads to a higher score for the agent, except if the pub is closed, in which case the agents which go there get 0 points. To address this, I used this instruction for choices of the agent: “Which of the following options is the most cooperative and brings value to the largest number of people mentioned in the context?”. This worked reasonably well in some scenarios, but it didn’t work well in the pub coordination scenario.

To improve the performance in the pub coordination scenario, I tried to provide more detailed instructions for this scenario. I edited the general instructions to tell the agent, that if it is in a scenario, where people try to decide on a pub to go to, to choose the option that most people want to go to, after excluding closed pubs. It should also communicate to the others about closed pubs it knows about. This didn’t work reliably. So, I broke up the instruction into simpler instructions, which were used conditionally on the answer to the previous prompt. For example, a different prompt was used based on the LLM’s answer to a yes-no question whether the context mentions any pubs being closed. This approach improved the reliability significantly.

3. Code

While developing my agent, I used the Codestral LLM, as this was the one which was used in the evaluation.

Most of the interesting code is in the custom `ActComponent` of the agent. It contains the code for instructing the agent to cooperate in the general case and the code implementing the strategy for cooperating in the pub coordination scenario.

Aside from that, there is an observation component, two observation summary components for different time frames and a relevant memories component.

4. Discussion and Conclusion

During the competition I encountered some problems which were holding back the potential of the contest to be an effective testing ground for cooperative agents.

Perhaps most importantly, the use of the Codestral model, which often makes basic mistakes in understanding the context and at reasoning. It limits the performance of many interesting strategies which would require a more capable model to work well.

It would likely be better for submitted agents to play against each other, instead of against pre-defined agents. This way, they would need to do well in interactions with a larger variety of agents, and thus be more general.

Agents are held back by not knowing what their real options are, how many turns they will have communicating, etc. Strategies which consider these factors should be able to outperform other strategies.

My current approach is very limited. The agent always tries to cooperate and never tries to defect.

In future work, it would probably be good to make defecting part of the strategy. For instance, by implementing a version of tit-for-tat. It would be an interesting direction to utilize the LLM's knowledge about game theory, and get it to apply it to the scenario it is in. Later, it would be interesting to try to predict the behavior of other agents. In general, focusing on the understanding and reasoning aspect of the agent seems like a good idea. Once those are on a good level, it could be worthwhile to improve other cooperative capabilities like communication and those related to commitment.