Agent theory approach:

Reinforcement learning is a subtype of artificial intelligence which is based on the idea that a computer learn as humans do — through trial and error. It aims for computers to learn and improve from experience rather than being explicitly instructed.

In order for the computer to do this the Learning algorithms are mathematical tools implemented by the programmer which allow the agent to effectively conduct trial and error when performing a task. Learning algorithms interpret the rewards and punishments returned to the agent from the environment and use the feedback to improve the agent’s choices for the future.

In reinforcement learning, the agent faces a dilemma which is known as the exploration-exploitation tradeoff. At what point should the agent exploit options which the agent thinks to be the best rather than exploring options which have the potential to be better or worse (or vice-versa)?

This tradeoff plays into something known as the multi-armed bandit problem, which is how one should dedicate a fixed amount of resources to several different options when you can never be certain what will come of exploring each state(environment)

Therefore we decided to use an ***epsilon-greedy approach*** which selects the action with the highest estimated reward most of the time. The aim is to have a balance between exploration and exploitation. Exploration allows us to have some room for trying new things, sometimes contradicting what we have already learned. Using this learning algorithm, our agent can converge to the optimal strategy for whatever situation it’s trying to learn

**Agent model(code):**

There is two part to our agent which follows the Markov model.

The first part is learning from the experience and the second part is to try the best possible local choice. Explore possible options. Both of these aspects can be changed by changing the alpha and beta elements.

First part is Experiencing:

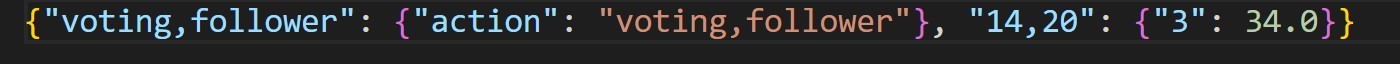
Text

Description automatically generated

After each move that the agent makes the actual result of that round will be calculated in the next round (After all the action in that round has been done). The difference between the two states will be calculated and added to the result calculated in the previous round. If the result was better, it will improve the result so it will be a possible option to choose if it was less, it will have a negative effect and reduce the result.

For example, in this case, choosing message three from a red agent and choosing potent message two from the blue human player caused a result better than expected. Therefore, some scores were added from the experience.

After first action:



After second action:



Text

Description automatically generated

IF the agent hasn’t been in this current state, it will choose the local optimum choice then it update the state and reward associated with that action and it will update the experience in the next round.

However, if the state has been visited. It checks the locally optimal choice. Then it will compare it with the result of all other actions. It will choose the highest result. This comparison is highly dependent on the alpha and omega coefficient that has been chosen in the Markov function. It leans towards what kind of agent we want. In our set up we chose reward and experience relatively similar, so the effect is almost the same.

Another factor we added to our code is when we chose the local optimum, if there is a similar result, we use random choice so it can explore different possibilities randomly.

Text

Description automatically generated

For example, in this state three actions cause the same state score. The third potent message was chosen randomly.

Text

Description automatically generated