Measurement of Novelty Difficulty in Monopoly

KMA Solaiman, Bharat Bhargava

Purdue University, West Lafayette, IN, USA
ksolaima@purdue.edu, bbshail@purdue.edu

Abstract

Novelties in learning algorithms are inherently different than anomalies and outliers. Novelty characterization and detection need to focus and differentiate between these terms. In recent times, DARPA Science of Artificial Intelligence and Learning for Open-world Novelty (SAIL-ON) project has shown advancement in novelty frameworks that discuss novelty characterization, detection and response for learning algorithms in perceptual and action domains. In this work, we implement one of the *Novelty Measurement* framework in a multi-agent multi-goal based agent domain. First, we breakdown the framework with the goal of implementation and testing for the Monopoly domain. Secondly, we propose a graph representation of the monopoly environment to be used as an effective representation of the agents mental model. Finally, we extend the learning framework of a reinforcement learning based agent to augment the state representation with the graph mental model. Experiments show the empirical calculation of the novelty difficulty according to the framework.

Motivation

(Alspector 2021) considered adaptation to novelty as a skill to learn to change by augmenting already existing skills for reacting to unfamiliar situations. They proposed the amount of edit to an effective representation in agent's mental models as a measure of difficulty for adaptation to novelty, defined as representation edit distance (RED). For the selection of an effective representation, they proposed different variants of representations i.e., neural nets, functions, or knowledge graphs. Our goal was to test this framework for adaptation and prediction of novelty in a multi-agent based domain - *Monopoly*. We breakdown the task into the following modules.

- Establish a representation for agent's mental model.
- Determine when the representation adequately describes the agent's mental model both in pre-novelty and in post-novelty to be considered as an effective representation.
- Include an optimization procedure for the agent to search for graph changes that could adapt to the generated novelties. Consequently, there should a method to detect the violation of expectation.
- Calculate the amount of edit that was needed for the effective representation to adapt to the novelty.

Monopoly is an adversarial multi-agent domain where agents have to make multiple decisions during the game. Even for a simple rule-based agent in Monopoly, the environment, rule, and action space is very large. Defining an effective representation for Monopoly, which could be edited for re-learning skills during introduction of a novelty is a difficult task. We used a combination of two variants of Monopoly agent to test this framework - 1) GNOME (Kejriwal and Thomas 2021), a rule-based fixed policy agent simulator, and 2) a deep reinforcement learning based agent (Haliem et al. 2021). Our initial intuition was - *a **concept graph** would be a good effective representation for agent's mental model in the monopoly environment*. The main contributions of this work are summarized as follows:

1. We breakdown an information theoretic framework for novelty measurement into multiple modules to empirically calculate the difficulty of novelty in an application domain, specifically *Monopoly*.
2. We proposed a methodology to build a graph representation of the Monopoly environment - *concept graph*. The graph is built on the smallest atomic units that can combine or derive to create new atomic units to deal with the novelties introduced in the application domain. We denote these smallest atomic units as *concepts*.
3. We compare the distance between the proposed graph representations before and after the novelty introduction for rule-based and reinforcement learning based agents in Monopoly, without any additional learning.
4. Finally, we propose a methodology to include the concept graph as a state representation for a reinforcement learning based monopoly agent, with the goal to learn the optimal representation of the Monopoly environment on which the edit distance can be computed as a metric of novelty difficulty.

Methodology

Concept Graph for Monopoly. We posit that the pre-novelty and post-novelty concept graphs are correlated. Following the idea of augmenting the existing skills in RED framework, the post-novelty graph has to be built on the derived concepts in pre-novelty graph. Concepts should be de-

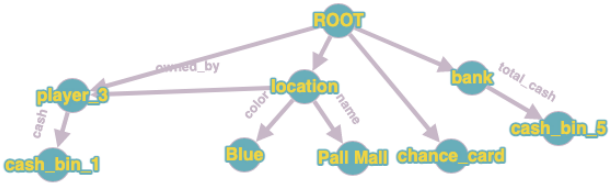

Figure 1: A snapshot of the Concept Graph for Monopoly.

composed into smaller atomic units that can create new concepts for the post-novelty world. Following this criterion, the representation would be defined as a pseudo-optimal representation. To begin with, we consider the concept graph to be a view of the environment from the agent's perspective. For monopoly, the world space and observation space can be considered to be the same in this environment (Boult et al. 2021). In (Kejriwal and Thomas 2021), the monopoly environment is described as a key-value schema. We build our concept graph from the tree-view of the schema. We have improved on the concepts derived from the schema for decomposing to smaller atomic units. In the simulator, the board is initialized with primary objects in the environment, i.e., location, agents, players, dice, bank, cards. Each of the primary object has attributes. For example, location has attributes such as, *owner, is mortgaged, owned by, number of houses/hotels*. In the concept graph, the primary objects become the nodes, while the attributes are relations, and the attribute values become the leaf nodes. Most attributes in the environment have discrete values. Cash, rent and property prices are continuous values and needed to be binned. Figure 1 shows part of the concept graph that we built for the game. Values of the nodes change as the game progresses.

Since monopoly is a multi-agent decision optimization problem, we choose to pick one agent as the perceptual agent and the other agents are recorded as part of the environment configuration for the perceptual agent. Since we are using graph as initial representation, according to the definition in (Alspector 2021), we chose graph edit distance (GED) as the RED approximation method. For the simulator in (Kejriwal and Thomas 2021), novelties are introduced during a tournament, not during the game. So, for computation of novelty difficulty, we compare between pre-novelty and post-novelty tournament outcomes where each tournament consists of N number of games.

Comparison of Representations without Learning. Initially we wanted to compare the trend change in the representation edit distance for Monopoly agent's mental model with the introduction of different novelties, irrespective of the representation being effective. Following the novelty introduction policy in (Kejriwal and Thomas 2021), we have two different concept graphs to compare for the representation edit distance in a rule-based fixed policy agent framework. There are two different design choices that we had to make before we compared the pre-novelty vs post-novelty representations.

- During a tournament, there are multiple turns for each player until one of the players monopolize. This creates multiple graph representations of the environment. So we needed to choose which representation we will compare.
- Since it is a multi-agent environment, we needed to decide which agents' representation change will be detected. We fix the random seed that controls the randomness during the simulation for pre-novelty and post-novelty games, and choose the same agent in both cases to compare.

Problem Description. Let us consider G_{pre} and G_{post} to be the pre-novelty and post-novelty graph representations, respectively. There are m number of turns in a pre-novelty game and n number of turns in a post-novelty game. So pre-novelty mental models are $G^1_{pre}, G^2_{pre}, \ldots G^m_{pre}$, and post-novelty models are $G^1_{post}, G^2_{post}, \ldots G^n_{post}$. At each turn during a single game as the concept graph changes according to the current values of the concepts. There are two types of change that takes place - static and dynamic. Current player cash or dice sequence are some of the dynamic properties that changes through each game turn. Property rent/ price, or location color are static concepts in the graph. We compare the GED between pre-novelty graph and post-novelty graph representation at each turn k, i.e., G^k_{pre} vs G^k_{post}, where $k \leq min(m, n)$. If we have two novelties, $novelty_1$ and $novelty_2$, and by the difficulty metric specified in the respective domain, if $novelty_1$ is harder to adapt than $novelty_2$, then the expected outcome is $(G^k_{pre}$ vs $G^k_{post})_{novelty_1} < (G^k_{pre}$ vs $G^k_{post})_{novelty_2}$. We present the results of this experiment in Figure 2.

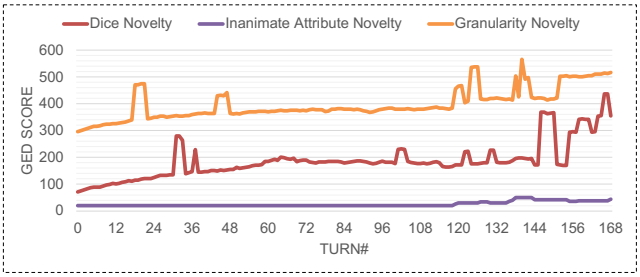

Figure 2: GED Scores between concept graphs from turns in a pre-novelty and a post-novelty Monopoly game.

Since Monopoly is a decision problem, rather than a classification problem and at each turn the agent decides on the best possible action, encoding the action in the concept graph should more clearly reflect the differences between pre-novelty and post-novelty environments. So we propose another variant of the concept graph where each time-step would be nodes and each edge would be an action that the agent takes. The final variant of the concept graph will encode both the environment and the actions of the agent. The agent would become the root with two edges - environment, and actions. Each branch would consist of the environment and action graph described previously.

Learning the Optimal Representation. The framework in (Alspector 2021) needs the representation to adequately describe the agent's mental model. We posit that it would be approximate to the concept graph for Monopoly, with the agent learning the best decision during game-play, and as a result learning the representation. In (Haliem et al. 2021), the agent is introduced as a Deep Q-network (DQN) based RL agent that learns winning strategies for Monopoly against different fixed policy agents. They design the game as a markov decision process (MDP) defined by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} >$ where $\mathcal{S}$ is the set of all possible states and $\mathcal{A}$ is the set of all possible actions. The transition function $\mathcal{T}$ is the probability that an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ will lead to a transition to state $s' \in \mathcal{S}$. The reward function $\mathcal{R}$ defines the immediate reward that an agent would receive after executing action a resulting in a transition from state s to s'. For state space representation, (Haliem et al. 2021) used a 240-dimensional vector where 16 dimensions are for the player representation and 224 dimensions are for the property representation.

Concept Graph as State Space for DQN based Monopoly Agent. For encoding graphs as state representation, first, we encode the node of each graph into a vector, encoding the structural properties around each node. To compute graph-level embedding, we aggregate the node-level embedding using the average or the attention strategy. The final graph embedding is passed as state value in the experience tensors of the DQN agent, as shown in Figure 3.

Among the existing state-of-the-art approaches for graph neighbor aggregation, we adopt Graph Convolutional Networks (GCN) (Kipf and Welling 2016), because it learns an aggregation function that is representation-invariant and can be applied to unseen nodes. GCN calculates the graph convolution from the representation of a node, u_n. In Figure 3, different colors represent different node types, and initially we represent the nodes with the concatenation of the one-hot-encoding of attribute values and node types. Before running the game with DQN agent, we simulate the game with rule-based-agents and collect the game log to create a dictionary of all possible attribute values. Finally, the nodes are initially represented with a 1059-dimensional vector. After multiple layers of GCNs, the node embeddings are ready to be fed into the Attention module (Att). After passing through the final GCN layer, the node embeddings are mapped to the dimensions of the filter size of the final layer. In our experiments, we use a final filter size of 32. So from 1059-dimensional vector, we get a 32-dimensional node embedding. We used the same global context-aware attention mechanism as described in (Bai et al. 2019) to gather the graph embedding.

Metrics of Effectiveness. After we have included the agent's mental model as part of the state-action-reward learning of the DQN agent, we needed to decide when a representation becomes effective post-novelty. The threshold for adequate post-novelty models should be chosen according to the learning model. The objective functions in a learning model reflect the performance metric which is used to determine the better model. For example, in Monopoly, the percentage of wins against other agents decide which is the better learning agent. For monopoly, we can either use the percentage of wins against other agents, or the number of games for pre-novelty convergence to determine a post-novelty model to be effective.

Experiments

Comparison of Representations without Learning. We introduce three different novelties in the post-novelty game according to the public simulator [1] introduced in (Kejriwal and Thomas 2021) -

1. Dice novelty - change in number of dices and dice state,
2. Inanimate attribute novelty - change in rent and property values,
3. Granularity novelty - change in location position, start sequence, and end sequence.

In Figure 2, each novelty follows a similar trend as the representations change with each turn. But there is a difference between the GED scores. Each novelty follow the same pre-novelty representation. According to that, inanimate-attribute novelty is the easiest among the three, where granularity novelty is the hardest to adapt.

Concept Graph as State Space for DQN based Monopoly Agent. We have ran initial experiments using the RL-based agent in (Haliem et al. 2021) where the state space is replaced with the proposed model in Figure 3. We played the RL-based agent against other rule-based agents. In Figure 4, we used attention network to aggregate node embedding to graph embedding with 3000 episodes for the tournament, whereas in Figure 5, we used the average of the node embedding values to calculate the graph embedding with 6000 episodes for the tournament. In both aggregation methods used in Figure 4 and 5, we can see irregular results for average Q-values and average reward. Initially, the performance was better than the standard DRL agent in (Haliem et al. 2021). Although, initially the model was learning as seen with the win rates, the performance worsens after a few thousands of episodes. We will present model improvements in future works.

Conclusion and Future Work

We have proposed a theoretical foundation and an empirical method for showing the efficacy of the information-theoretic framework proposed in (Alspector 2021). For achieving that goal in Monopoly domain, we have proposed a methodology to build and use concept graph as a state representation for a DQN agent. The architecture for learning the concept graph as the optimal representation of the monopoly environment has not achieved convergence yet. Application of the framework introduced in (Alspector 2021) to measure novelty to a practical application domain is not a straight forward method. It requires following and applying the step-wise techniques introduced in this work to the specific domain.

[1] https://github.com/mayankkejriwal/GNOME-p3

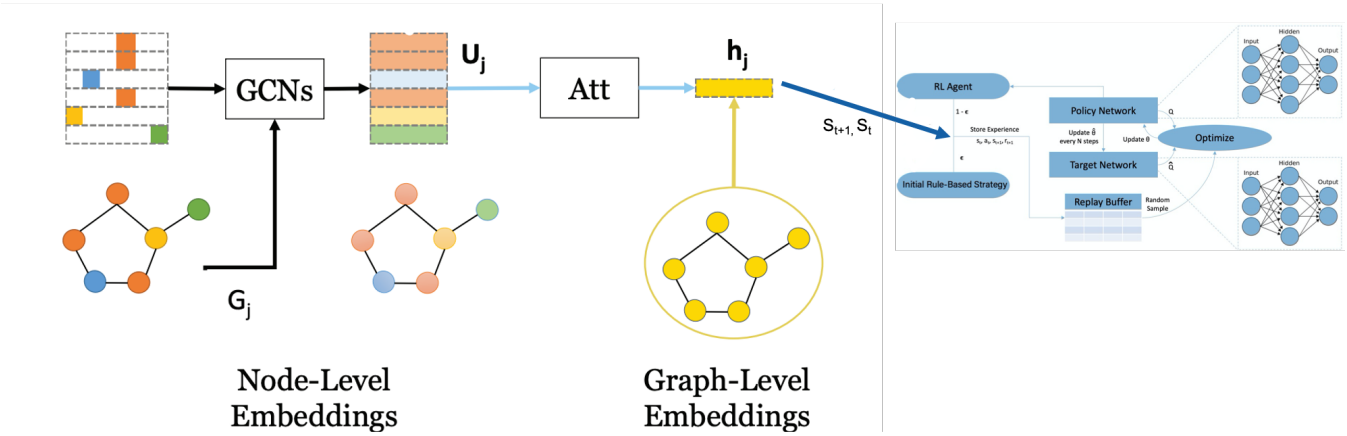

Figure 3: Concept Graph Embedding as State Representation for DQN-based Monopoly Agent.

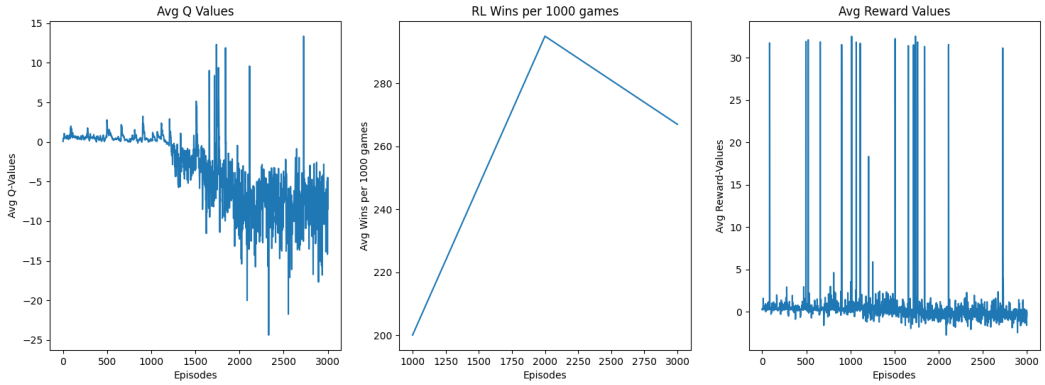

Figure 4: Results for Concept Graph as State Space in Monopoly DQN Agent with Attention Network.

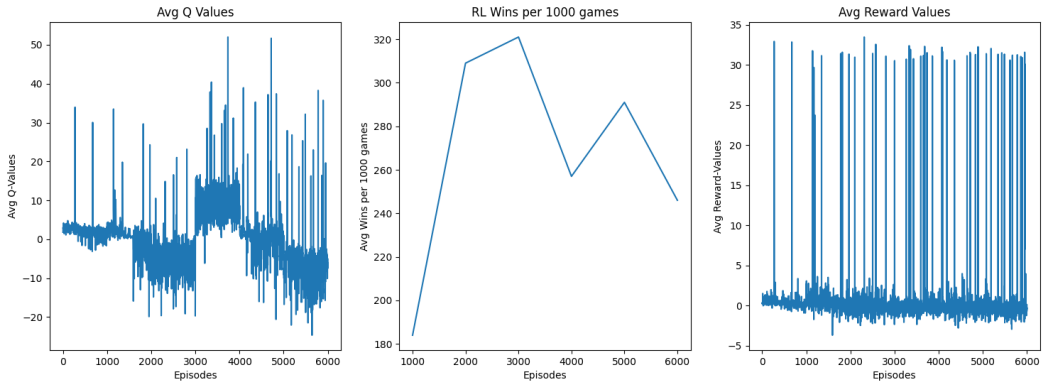

Figure 5: Results for Concept Graph as State Space in Monopoly DQN Agent with average aggregation.

In future, we would need to include an optimization procedure for the agent to search for graph changes for adapting to the generated novelties. This would include finding a search method to detect the violation of expectation. Finally, we will evaluate the string edit distance on the learned vector representations or the graph edit distance on the concept graph to calculate the difficulty of the generated novelties.

References

Alspector, J. 2021. Representation Edit Distance as a Measure of Novelty. arXiv:2111.02770.

Bai, Y.; Ding, H.; Bian, S.; Chen, T.; Sun, Y.; and Wang, W. 2019. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 384–392.

Boult, T.; Grabowicz, P.; Prijatelj, D.; Stern, R.; Holder, L.; Alspector, J.; Jafarzadeh, M.; Ahmad, T.; Dhamija, A.; Li, C.; et al. 2021. Towards a Unifying Framework for Formal Theories of Novelty. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 15047–15052.

Haliem, M.; Bonjour, T.; Alsalem, A.; Thomas, S.; Li, H.; Aggarwal, V.; Bhargava, B.; and Kejriwal, M. 2021. Decision Making in Monopoly using a Hybrid Deep Reinforcement Learning Approach. arXiv:2103.00683.

Kejriwal, M.; and Thomas, S. 2021. A multi-agent simulator for generating novelty in monopoly. *Simulation Modelling Practice and Theory*, 112: 102364.

Kipf, T. N.; and Welling, M. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *CoRR*, abs / 1609.02907.

Acknowledgments

This research is supported, in part, by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under the contract number W911NF2020003. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, AFRL, or the U.S. Government. We thank our team members on this project for all the discussions to develop this paper. Some of the ideas in this paper are based on our learning from the SAIL-ON meetings.