



# Optimizing a Gamified Design Through Reinforcement Learning - A Case Study in Stack Overflow

Jonathan Martin <sup>1</sup>[0000-0003-0450-9480], Diego Torres  
<sup>1,2</sup>[0000-0001-7533-0133], and Alejandro Fernandez<sup>1</sup>[0000-0002-7968-6871]

<sup>1</sup> LIFIA, CICPBA-Facultad de Informática, UNLP, Argentina  
{jonathan.martin, diego.torres,  
alejandro.fernandez}@lifia.info.unlp.edu.ar  
<sup>2</sup> Depto. CyT, UNQ, Argentina

**Abstract.** Gamification can be used to foster participation in knowledge sharing communities. While designing and assessing the potential impact of a gamification design in such a context, it is important to avoid work disruption and negative side effects. A gamification optimization approach implemented with deep reinforcement learning based on play-testing approaches helps prevent possible disruptive configuration and has the capability to adapt to different communities or gamification targets. In this research, a case of study for this approach is presented running over the Stack Overflow Q&A community. The approach detects the best configuration for a Contribution, Reinforcement, and Dissemination (CRD) gamification strategy using Stack Overflow historical data in a year. The results show that the approach finds proper gamification strategy configurations. Moreover, those configurations are robust enough to be applied along the time unseen periods.

**Keywords:** Deep Reinforcement Learning · Gamification · Knowledge building community · Optimization · Stack Overflow

## 1 Introduction

Stack Overflow is a well-known questions and answers (Q&A) community with a large number of users [2], more precisely with 12,615,110 registered users<sup>3</sup>. Stack Overflow covers the knowledge construction process from a question and answers perspective. Each user can vary between Question Author and Answer Author's roles to meet the community's requirements.

In order to lead the users to fulfill this role is important to encourage participation in the community. There are different approaches to encourage participation in a Q&A community. The use of Gamification [16] to encourage participation consists of applying elements taken from the realm of game and videogames to non-ludic environments without modifying its central structure [8, 3, 10].

---

<sup>3</sup> Information obtained on 25/06/20 at  
<https://data.stackexchange.com/stackoverflow/query/1255610>

Stack Overflow already implements a gamification strategy that extends PBL (points, badges, and leader boards) with other gamification elements [7]. Those gamification strategies, as PBL, are an effective method to build and sustain a productive and active Q&A community directing the participation towards behaviors that improve the quality of content [6, 8].

The result of applying a gamification approach to a community cannot be predicted entirely [18, 14, 9], and no guarantee that the desired effect will be achieved. It can generate disruptions to the community, create unproductive competition patterns between community members, or hinder participation [10, 25]. An example of this is a design that promotes individual competition in a collaborative context due that collaboration should be the most relevant value [21].

If not done carefully, the gamification changes can disrupt the normal development of the activity aimed to improve. Play-testing is a process used as part of the iterative process for game design, development, evaluation, monitoring, and adaptation [23]. It could be applied to this context, but changes to the gamification can damage or fail to be adopted by the community. Another difficulty from play-testing is that it requires a functioning game, players, and time.

Other approaches consist of optimization strategies used to reduce the effort/cost of creating an effective gamification design where optimal pseudo-rewards guide the users in the activity [17]. However, this approach requires absolute knowledge of the decision environment.

This paper proposes as an alternative an approach to optimize the configuration of a gamified design, using a reinforcement learning agent to explore the space of configuration possibilities as in a play-testing approach. This approach would allow optimizing a configuration without disturbing the community and with fewer requirements of the traditional play-testing approach.

Consequently, a Contribution, Reinforcement, and Dissemination (CRD) strategy for gamified design optimization in knowledge-building communities is presented, using a reinforcement learning (RL) agent to explore the space of configuration possibilities. The agents look for configurations that reward desired user behaviors for the selected pair community/reward.

We evaluate the approach in a sub-community of Stack Overflow for one year. The evaluation was developed in 2 stages: a training stage in the first six months; and then a second evaluation stage in the following six months. Badges and ranks are the principal gamification components for both scenarios. The results show the stability of configurations obtained by this method during different time-lapses.

The rest of this article is organized as follows: Section 2 explore the related work; Section 3 presents a gamified design that serves as the basis for the rest of this work; Section 4 provides an overview of our approach; then, Section 5 introduces the evaluation over the dataset extracted from Stack Overflow. The results are described in Section 6, and they are discussed in Section 7. Finally, the Conclusions and further work are detailed in Section 8.

## 2 Related Work

Lieder et al. [17] explore ways to obtain an optimal gamification design by a mathematical framework. They introduce a gamification approach for problems that could be modeled as a Markov decision process; they compute optimal pseudo-rewards to guide the users in the activity for what is requested perfect knowledge of the decision environment or the possibility of approximate it. Our work achieves an “optimal gamification” but from an existing gamification design.

With an existing gamification design, the way to achieve an “optimal gamification” is to configure it to fit our purposes. This approach addresses the system configuration problem topic. The problem typically involves learning from analyzing actual executions or historical data, model specific aspects of the systems, and then adapt to actual conditions based on requirements [12]. A common way of tuning configurations is done manually by performance engineers, spending several hours of work [1]. There were proposed other methods less time-consuming and more precise to find configurations.

A rule-based approach like Multirelational Data Mining (MRDM) helps discover patterns. However, it requests gathering the metadata from the database of the system to configure, which describes the best approach of the analysis and transformation of the database into MRDM formats [22, 4]. This approach favors quickly finding a suitable configuration at the expense of optimality.

A model-based approach is concerned with conducting experiments on a chosen set of configurations to observe their performance [11, 30, 26, 15, 24]. The restriction of using a limited number of configurations is a limitation that our approach overcomes because there are numerous configurations in practice with many dependencies between them and these can be used to get better optimization. Our approach instead evaluate all parameters and process them jointly over an RL process in order to exploit them.

Search-based approaches begin with an initial configuration to perform sequential experiments but require a statistical model to fit [30, 31, 4, 19]. This kind of model is similar to the Lieder et al. model presented above and requires full knowledge of the decision environment.

Finally, as proposed in this work, learning-based approaches find the optimal configuration by reacting to feedback [32, 5]. The RL approach used here fits better to our context where there are no correct input/output pairs required for other learning approaches that defined a correct configuration of gamification. However, a policy of satisfaction could be defined for our unknown environment by a reward function that measures the configuration found.

## 3 CRD Gamification Design

This work uses a Contribution - Reinforcement - Dissemination (CRD) gamification design based on Metagame [18], a knowledge-building CRD gamification design. CRD defines a PBL gamification strategy based on four main types of

actions (inputs): Contribution, to represents the creation of content; Reinforcements, for content editing, data retrieval, status changes, or content groupings; Dissemination for actions linked to spreading the content, for example on social networks or within the community itself; and finally, Loggin actions.

The players (the users of the application) are awarded badges when they provide a certain number of inputs. There is one badge for each class of input, and there are multiple levels for each badge. To reach a higher level of a badge, more inputs of the corresponding class are required.

Players are also ranked according to the badges they obtain. For example, they start as “Visitors” (Rank 1), and they become “Explorers” (Rank 2) when they obtain the Login level 2 badge. After they obtained one “Contribution” badge, one “Reinforcement” badge, and one “Dissemination” badge, they become “Editors” (Rank 3)<sup>4</sup>. Then, after they obtain ten badges, they become “Prolific editors” (Rank 4). Finally, as long as they earn a new badge every month, they obtain and maintain the level of “Committed editors” (Rank 5).

The amount and type of badges a player requires to be promoted from one rank to another is part of the gamification configuration. How many inputs do users need to perform to get the 3rd level of Contribution badge? If the configuration sets a large number of inputs for promoting one badge, how does this impact the game’s evolution? Will players be comfortable with this difficulty level?

In this article, the configuration space is reduced to five variables: the number of levels per badge; and, for each badge type, the number of inputs required to pass from one level to the next.

The simulation of the gamification design runs over a slice of historical data from the community. This selected slice becomes the simulation dataset.

The challenge is to find the correct values for the five variables of the configuration and then test those values in a “simulation of the game” with a historical dataset. The simulation is to avoid a harmful configuration that threatens the actual community.

The simulated game awards a badge to the users and computes the ranking. In an iterative process, users’ distribution in the ranking could be analyzed to perform a fine-tune to the configuration. It could be repeated until a suitable configuration is reached (e.g., all ranks have users.).

There are two main issues in this approach. First, it is time-consuming; even with only five variables to set, the combined possibilities are many, and each change requests simulation with its analysis of the results. Secondly, it is based on the researchers’ intuition regarding the effect that change in the configuration may have on the user’s attitudes. The following section introduces an alternative approach that uses a reinforcement learning agent to explore the configuration space effectively.

---

<sup>4</sup> In the original work, they were called Citizen Scientists; we have changed the term for the sake of clarity

## 4 Gamification Configuration with RL: Historical RL Framework

Reinforcement Learning (RL) is an approach to train machine learning models to make sequences of decisions. In RL, an agent takes actions in an attempt to maximize a cumulative reward [28]. Exploring the configuration space to find an acceptable configuration for the gamified design involves multiple iterations of updating the configuration, simulating the game on the defined dataset, and observing the effect. Having an agent instead of a person in charge of the exploration allows us to explore more alternatives in less time.

Figure 1 provides an overview of our approach to find an optimal game configuration using an RL agent. The game’s configuration is modeled as an N-dimensional vector (1) of positive numbers. The game is simulated by providing a configuration and a slice of the community history (2) to the game engine (3). Once the simulation finishes, the list of badge assignments and users’ ranking (4) is passed back to the agent. At each step (5), the RL agent (6) changes only one dimension of the configuration vector, by one, up or down. Then, the simulation runs. The agent uses a reward function (7) to update its model based on its last action outcomes. In one episode, the agent performs a maximum of 300 steps until it considers no further changes to the configuration are useful, then stops.

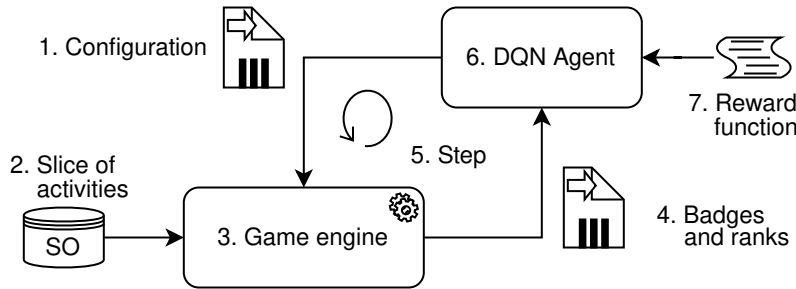


Fig. 1. Approach overview

This work uses a Deep Q-Network (DQN) [20], a reinforcement learning agent that uses a deep convolutional neural network to learn successful policies in high-dimensional state spaces like video games or robotics. Due to that, the DQN agent can handle complex, high-dimensional spaces. This means that it can deal with an even richer (and more complex) game configuration. DQN has become popular in the last few years due to the increasing availability of ready-to-use libraries (such as DeepMind’s implementation for the Lua language) and services (such as OpenAI API). However, the type of agent could be configured according to the use case and analysis.

DQN is an implementation of Q-learning. Q-learning is a form of model-free reinforcement learning, which can be viewed as a method of asynchronous

dynamic programming (DP). Learning proceeds in a similar way to Sutton’s method of temporal differences (TD) [27]: an agent tries an action in a particular state and evaluates its consequences in terms of the immediate reward or penalty. By repeatedly trying all actions in all states, it learns by evaluating the long-term reward.

Although Q-learning is a very powerful algorithm, its main weakness is its lack of generality. If Q-learning is viewed as the updating of numbers in a two-dimensional array ( $A * S$ ), it is, in fact, similar to dynamic programming. This indicates that for states that the Q-learning agent has not seen before, it has no idea what action to take. In other words, the Q-learning agent does not have the ability to estimate the value of unseen states. To deal with this problem, DQN discards the two-dimensional matrix by replacing it with a Neural Network.

Two other techniques are also essential for DQN training:

**Experience replay:** Since the training samples in the typical RL setup are highly correlated, and less data-efficient, this will lead to higher convergence of the network. One way to solve the sample distribution problem is to adopt experience replay. Essentially, sample transitions are stored, which will then be randomly selected from the “transition pool” to update the agent’s knowledge.

**Separate target networks:** The Target Network  $Q$  has the same structure as the one that estimates the value. Each  $C$  step, the target network is readjusted to the other. Therefore, the fluctuation becomes less severe, resulting in more stable training.

Those techniques were applied to the agent used in this approach to obtain an optimized configuration for the gamification. Also, the five-variable configuration discussed in Section 3, implies that the difficulty of reaching level  $x + 1$  after obtaining the badge of level  $x$ , is comparable to the difficulty of obtaining the badge of level  $x + 2$ , after being awarded the badge of level  $x + 1$ . Such a linear relationship among subsequent challenges harms user engagement. To introduce variability, we define  $inputs(x)$  as the number of inputs required to obtain a badge of level  $x$  (see Equation 1). The larger the values of variables  $b$  and  $m$ , the greater the steps between levels. The values of  $m_{oc}$  and  $b_{oc}$  define the direction of those steps. Therefore, the configuration consists of 17 dimensions, the number of levels per badge (all 4 badges have the same number of levels), and values for the variables  $m$ ,  $b$ ,  $m_{oc}$  and  $b_{oc}$  for each of the four badges.

$$inputs(x) = \max(\sin(m_{oc} \times x + b_{oc}) \times (m \times x + b), 1) \quad (1)$$

Following this approach, at the start of each reinforcement learning step, the agent produces a new configuration. The game is simulated using that configuration, on a predefined simulation dataset. Then, the reward function is applied to evaluate how well it did (i.e., is the resulting configuration better?) and notifies the agent about the performance. This process continues for a predefined number of episodes.

To enable the coupling of Stack Overflow to the simulator, we must detect the actions related to the gamification, obtain this history of actions, and finally integrate them into the simulation flow. Table 1 presents the mapping of Stack Overflow actions to CRD gamification actions.

**Table 1.** Stack Overflow events in the gamification context

| <b>Contribution</b>  | <b>Reinforcement</b>                  | <b>Dissemination</b>                       |
|----------------------|---------------------------------------|--|
| Question Creation    | Question Editing                      | Tag Creation                               |
| Creating answers     | Editing answers                       | Editing tags                               |
| Creation of comments | Rollback of questions                 | Rollback of tags                           |
| Post Protection      | Response Rollback                     | Post Migration                             |
| Locking posts        | Changing post statuses                | Creating links                             |
|                      | Merging of questions                  | Discussion of comments                     |
|                      | Making of suggestions for change      | Tweeting of posts                          |
|                      | Application of suggestions for change | Question marked as a trend                 |
|                      |                                       | Question highlighted by the user community |

## 5 Evaluation

The evaluation aims to evaluate the approach to obtain an optimal configuration for a gamification design in knowledge-building communities through reinforcement learning, introduced in this article. The evaluation is structured answering the following questions:

- **Q1:** How is the configuration obtained? The detail of variables and the final reward function.
- **Q2:** Is the difficulty of the gamification achievable and enjoyable by the player?. We followed the gamification flow aspects related to *challenge* and *player skill* [29, 13]. In a minimalist way, the difficulty should change along the time and adapted to the skills the user incorporates.
- **Q3:** There are correlations among the number of badges the configuration assign?
- **Q4:** Could users reasonably achieve ranks?
- **Q5:** Is the detected configuration robust when it is applied with new community activities? How is the behavior in terms of the former questions? The goal here is to analyze the robustness of the configurations obtained by the agent’s adjustments in the same community beyond the period known by the agent.

Therefore, the evaluation stages were divided into 2 stages: A training stage in the **first semester**; and a second evaluation stage in the **second semester** using the previous configuration.

### 5.1 Materials

The dataset for the evaluation was taken from the Python community from Stack Overflow (filtered by the *Tags* property with the value “python”) of 2018. This

translates to 242,822 questions, their answers, comments, and activity histories which result in 3,337,788 gamification actions. Among these actions, the Login actions were selected as the beginning of action’s groups by the *RevisionGUID* tag. This dataset was divided into two parts of 6 months each: from January to June the first, and from July to December the second. This division was made with the goal of training the agent in the first half and evaluating its performance in the second part of the dataset in order to answer question Q5.

Finally, the Equation 2 presents the reward function used to train the agent. The reward function drives the agent to look for configurations that value reinforcement inputs over contribution inputs, and dissemination inputs over both contribution and reinforcement inputs. The choice of this function was due to the fact that, as mentioned above, it rewards the behavior that we consider relevant in a knowledge building community. Also, this function will help in the evaluation of the result of applying the configuration obtained from the training in the test dataset.

$$reward = \sum_{u \in Users} rank(u) \times (ci(u) + ri(u) \times 5 + di(u) \times 10) \quad (2)$$

where:

$U$  = users that provided input to a featured article.

$rank(u)$  = the rank (1..5) of user  $u$  in the simulated gamified design.

$ci(u)$  = count of contribution inputs of user  $u$

$ri(u)$  = count of reinforcement inputs of user  $u$

$di(u)$  = count of dissemination inputs of user  $u$

## 6 Results

### 6.1 First semester

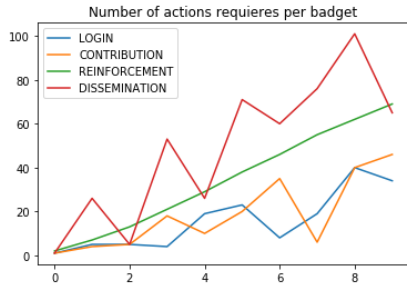
During the training process, the RL agent was trained using the first semester of the data set. The agent was trained for 100 episodes, obtaining a maximum score of 7,445,496 points awarded by the reward in combination with the dataset. Answering **Q1**, the agent returns a configuration with 10 badges per type with the parameters presented in Table 2. Dissemination receives the highest values in all of the variables representing high values for the oscillation ( $m_{oc}$  and  $b_{oc}$ ) and steps bigger than the other badges.

In order to answer **Q2**, Figure 2 shows a difficulty analysis by comparing the number of new actions required to obtain each badge by action type. Badge levels are on the  $x$  axis, and the  $y$  axis indicates the number of inputs required to earn a badge. For example, a player has to perform 25 dissemination actions to achieve level 1 of the *Dissemination badge* (the line grows up from 0 to 1), then the same player needs only 7 new dissemination actions to be promoted to level 2 of *Dissemination badge* (the line decreases from 1 to 2). These oscillations in the lines in Figure 2 means that the effort of the player to get a new badge is

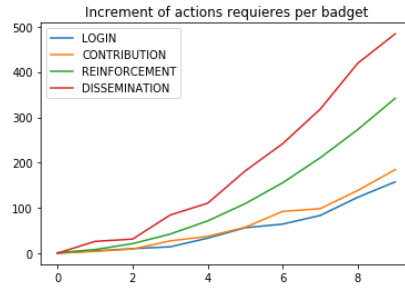


**Table 2.** Configuration for python community, first half of 2018.

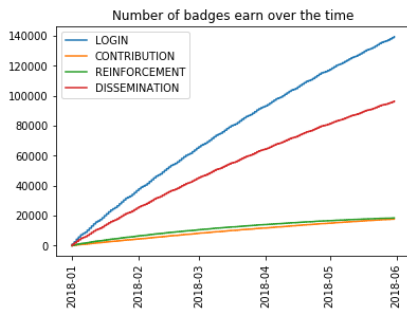
| Action type   | #max_badges | $m_{oc}$ | $b_{oc}$ | $m$ | $b$ |
|---------------|-------------|----------|----------|-----|-----|
| Contribution  | 10          | 2        | 5        | 6   | 1   |
| Reinforcement | 10          | 3        | 9        | 7   | 7   |
| Dissemination | 10          | 14       | 19       | 14  | 13  |
| Login         | 10          | 4        | 4        | 5   | 1   |



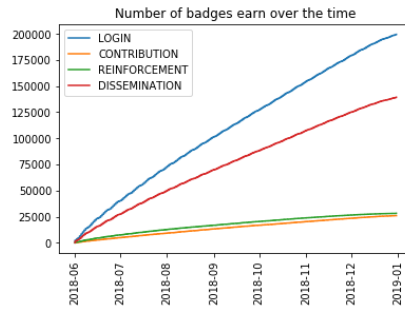
**Fig. 2.** Difficulty analysis: New actions required to obtain each badge.



**Fig. 3.** Total actions required to earn each badge.



**Fig. 4.** First semester. Awarding of badges to users over time.



**Fig. 5.** Second semester - Legacy Configuration. Delivery of badges to users over time.

not always the same. The player had hard working periods with a lot of actions (challenges) and then a period of skills assimilation, in the line with having a balanced flow as we have mentioned in **Q2**. However, in the case of Reinforcement badges, the configuration proposed to have a lineal increment of difficulty along with the game.

The scaling of the difficulty could be seen in Figure 3 that shows the accumulation of actions required to obtain each badge. As in the previous Figure, the  $x$  axis represents the badge levels and the  $y$  axis denotes the total of inputs required from the beginning to earn some badge. In this Figure, is possible to see a difference between the difficulty to earn a dissemination and reinforcement badges in comparison to the other badges; they become more challenging, at a faster pace than the others. It is because these actions are the main target of the reward and also the number of reinforcement actions are relevant elements in the activity of the community’s users.

To answer **Q3**, Figure 4 shows how the game would have assigned badges with this dataset and configuration. The  $x$  axis is the time along the first six months of data, and the  $y$  axis, the number of delivered badges. For example, in February 2018, the game delivered near than 20.000 Dissemination badges. As we can see, there is a correlation between reinforcement and contributions because both lines grow with similar values. As a complement of the former Figure, all players can achieve all the challenges to obtain the badges.

Finally, to answer **Q4**, Table 3 shows the distribution of players in each of the ranks with both the actual number and the percentage representation of this number over the total number of users in this dataset. From this data, we can observe the presence of users in all ranks and a coherent proportion of them: the distribution of the ranks is similar a pyramid with more presence in the easier ranks and a small group of players in the hardest rank.

**Table 3.** Rank of players in python community, first half of 2018.

|                   | <b>Rank 1</b> | <b>Rank 2</b> | <b>Rank 3</b> | <b>Rank 4</b> | <b>Rank 5</b> |
|-------------------|---------------|---------------|---------------|---------------|---------------|
| <b>Real</b>       | 61129         | 5084          | 3658          | 332           | 1446          |
| <b>Percentage</b> | 85.31%        | 7.09%         | 5.1%          | 0.46%         | 2.01%         |

## 6.2 Second semester - Legacy configuration

In order to answer **Q5**, the agent applies the configuration obtained to the second half of 2018. (Table 2). As an evaluation metric, the reward value given to this configuration over the simulation was considered; this value was 11,862,732 points. Although the value is significantly higher, this is due to the fact that the activity in the second half of the year is also significantly higher. So these reward values are also highly correlated to the simulation dataset.

As in the first half, information related to the difficulty, scaling is the same due to its relation with the configuration. Instead, the badge distribution in-

creases significantly as it's shown in Figure 5. This is related to the increment of user activity with respect to the previous dataset. However, the distribution of badges looks similar in both datasets and it means that the configuration has robustness for the same community along the time.

The Table 4 is presented showing the distribution of players in each of the ranks. These data show an expected proportion due to the continuation over time of the dataset. Along with this, we can observe that the increase in reward values is related to the number of players per rank, being almost double in the higher ranks, although in reality, the proportion between the two semesters is almost the same. It means that the configuration generated by the agents fits correctly beyond the period known by the agent.

**Table 4.** Range of players in python community, second half of 2018 with legacy configuration.

|                   | <b>Rank 1</b> | <b>Rank 2</b> | <b>Rank 3</b> | <b>Rank 4</b> | <b>Rank 5</b> |
|-------------------|---------------|---------------|---------------|---------------|---------------|
| <b>Real</b>       | 86090         | 7365          | 5483          | 658           | 2125          |
| <b>Percentage</b> | 84.63%        | 7.24%         | 5.39%         | 0.64%         | 2.08%         |

## 7 Discussion

This section presents an analysis of the results obtained in the application of the deep reinforcement learning approach to obtain an optimal configuration for a gamification design in the Python sub-community of Stack Overflow.

From the configuration obtained by the agent answering the **Q1**, we can observe a more pronounced separation into two groups of badges, dissemination, and reinforcement versus contribution and login. While among all the training a priority ordering of badges dictated by the reward function is preserved as much as possible, in these particular cases the impact of the reward function is further accentuated. Also, we can observe from the **Q2** and **Q3** answers, how the agent manages the difficulty curve of badges such as dissemination to allow an easy initial acquisition, thus providing access to the first ranks. While with badges such as the reinforcement badge, the difficulty curve is flattened at higher levels to aid in the preservation of the last rank with its temporary feature. These results show that the configuration found by the agent presents characteristics to be achievable and enjoyable by the player.

The first relevant difference found when working with the data divided into the first and second semesters for the **Q5** was based on the fact that the magnitudes of activity in both sets were different. The second-semester data set was larger than the first semester data set, and this could condition the performance of the agent trained in the first semester.

However, we were able to observe that the configuration obtained in the first semester turned out to be effective, although probably not optimal due it wasn't

optimized for this period, in the simulation of the second semester. This is due to the fact that, although the magnitudes are different and this will disturb the agent’s configuration process, the activity proportions and the form of user activity did not vary from one semester to another, so the configuration remained within acceptable margins.

These magnitude differences exist also in the distribution of users by ranks. But this difference didn’t generate a negative impact on the proportional distribution of users. The answer to the **Q4** shows that all the ranks were covered in a pyramid shape, except by the last rank. It is a temporal dependant rank and as it was a simulation, it is conditioned by the ending date of the simulation. In both rank distribution, there is a big part of the users that stay in the first rank. It’s common in communities where new users only participate one time to make a question and after never more. Table 5 shows a comparison of the ranks obtained with the strategy presented in this paper in contrast with a manual configuration of badges distribution from a previous paper [18]. In this table, an increment of players in higher ranks can be observed buy it still maintains its pyramidal shape, which means that the players are able to achieve high ranks.

**Table 5.** Ranks comparative

|                               | <b>Rank 1</b> | <b>Rank 2</b> | <b>Rank 3</b> | <b>Rank 4</b> | <b>Rank 5</b> |
|-------------------------------|---------------|---------------|---------------|---------------|---------------|
| <b>Previous approach [18]</b> | 96.2%         | 2.28%         | 1.52%         | 0.0%          | -             |
| <b>First half of 2018</b>     | 85.31%        | 7.09%         | 5.1%          | 0.46%         | 2.01%         |
| <b>Second half of 2018</b>    | 84.63%        | 7.24%         | 5.39%         | 0.64%         | 2.08%         |

Although the article is based on using a specific gamification design, other types of gamification design are an exciting aspect of research. In this case, we base the search for the optimal configuration for a badge-based configuration (5 variables detailed above). What does it imply to use a different gamification strategy with or without badges? What does happen if it requires a higher number of variables? As a first approach to use a different gamification strategy, we have to identify and map the user’s activities of the community with the actions of the gamification strategy. This mapping is required in order to have a simulation to optimize. Moreover, a new reward function based on the values of the new simulator has to be created. Also, it is necessary to identify the configuration variables and create methods that allow the RL agent to changes them. If a high number of variables is requested, the agent will have to learn the relevance of more actions and the changes that those made in the configuration, but the process presented in this approach will not change.

Another important aspect is to extend the approach to contemplate a gamification strategy in a tailored way. The overall logic and purpose of gamification are maintained, e.g., promoting collaborative work, quality content production, and dissemination. Furthermore, it optimizes the way players are motivated by adapting the gamification alternatives. Each player can give their best based on their behavioral profile in conjunction with the overall purpose.

## 8 Conclusions

This work introduced an approach to optimize a gamified design to reward desired user behaviors without disturbing the functionality of an existing community. This approach is based on play-testing using a deep reinforcement learning strategy.

The optimization was performed over a simulation of gamification by a deep reinforcement learning agent. It allows us to adapt the distribution of badges by a change in the parameters of the badges distribution function. Those parameters are defined by the reinforcement learning agent while it tries to optimize a reward function.

As an evaluation of the approach, the integration of the knowledge-building community, Stack Overflow, is presented. For this purpose, notions of the community's data structure were also introduced, as well as the data extraction and mapping process. Also, a reward function to encourage communication and dissemination in the community is developed to train the agent and as a metric of performance for the configuration obtained.

The configurations obtained by the agent during the training show an improvement over a previous manual configuration approach. This new approach helps to configure gamification that allows the players to flow over the ranks; without them having to change their main behavior in the community. The agent also detects and values correctly different types of actions by the importance of the reward function and its occurrence in the dataset. It helps to fit the difficulty of the gamification to the community and the target goal reducing the possibility of harming the community. It was proved that the configurations of a previous period can be successfully maintained along the time for the same community archiving similar badges and users in rank distributions.

As future work, given the sensitivity of the agent to the input data, it would be relevant to evaluate ways of overcoming this limitation or explore regularization strategies with respect to the input values. Then, try to optimize the configuration for the second dataset with the pretrained agent is the next logical step.

In order to facilitate this future work, it is necessary to consider the creation of a framework to evaluate and optimize the integration of gamification and communities. It will allow us to get a better abstraction between the agent and the environment for training. Finally, this abstraction will allow us to change or optimize the agent implementing novel deep reinforcement learning algorithms to obtain more accurate configuration in gamification with more complex configurations or action space. The capability of change easily the agent will allow comparing different RL agents for this specific task.

## References

1. Allen, S.T., Jankowski, M., Pathirana, P.: Storm Applied: Strategies for real-time event processing. Manning Publications Co. (2015)

2. Anderson, A., Huttenlocher, D., Kleinberg, J., Leskovec, J.: Discovering value from community activity on focused question answering sites: a case study of stack overflow. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 850–858 (2012)
3. Beza, E., Steinke, J., van Etten, J., Reidsma, P., Fadda, C., Mittra, S., Mathur, P., Kooistra, L.: What are the prospects for citizen science in agriculture? Evidence from three continents on motivation and mobile telephone use of resource-poor farmers. *PloS One* **12**(5), e0175700 (2017). <https://doi.org/10.1371/journal.pone.0175700>
4. Bilal, M., Canini, M.: Towards automatic parameter tuning of stream processing systems. In: Proceedings of the 2017 Symposium on Cloud Computing. pp. 189–200 (2017)
5. Bu, X., Rao, J., Xu, C.Z.: A reinforcement learning approach to online web systems auto-configuration. In: 2009 29th IEEE International Conference on Distributed Computing Systems. pp. 2–11. IEEE (2009)
6. Burke, B.: Gamification 2020: What is the future of gamification. Gartner, Inc., Nov **5** (2012)
7. Cavusoglu, H., Li, Z., Huang, K.W.: Can gamification motivate voluntary contributions? the case of stackoverflow q&a community. In: Proceedings of the 18th ACM conference companion on computer supported cooperative work & social computing. pp. 171–174 (2015)
8. Deterding, S., Dixon, D., Khaled, R., Nacke, L.: From game design elements to gamefulness: Defining "gamification". In: Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments. pp. 9–15. MindTrek '11, ACM, New York, NY, USA (2011). <https://doi.org/10.1145/2181037.2181040>
9. Devers, C.J., Gurung, R.A.R.: Critical Perspective on Gamification in Education. In: Gamification in Education and Business, pp. 417–430. Springer International Publishing, Cham (2015)
10. Domínguez, A., Saenz-De-Navarrete, J., De-Marcos, L., Fernández-Sanz, L., Pagés, C., Martínez-Herráiz, J.J.: Gamifying learning experiences: Practical implications and outcomes. *Computers and Education* **63**, 380–392 (2013). <https://doi.org/10.1016/j.compedu.2012.12.020>
11. Fischer, L., Gao, S., Bernstein, A.: Machines tuning machines: Configuring distributed stream processors with bayesian optimization. In: 2015 IEEE International Conference on Cluster Computing. pp. 22–31. IEEE (2015)
12. Geldenhuys, M.K., Thamsen, L., Gontarskay, K.K., Lorenz, F., Kao, O.: Effectively testing system configurations of critical iot analytics pipelines. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 4157–4162 (Dec 2019). <https://doi.org/10.1109/BigData47090.2019.9005504>
13. Göbel, S., Wendel, V.: Personalization and adaptation. In: Serious Games, pp. 161–210. Springer (2016)
14. Hamari, J., Koivisto, J., Sarsa, H.: Does Gamification Work? – A Literature Review of Empirical Studies on Gamification. In: 2014 47th Hawaii International Conference on System Sciences. pp. 3025–3034 (2014). <https://doi.org/10.1109/HICSS.2014.377>
15. Jamshidi, P., Casale, G.: An uncertainty-aware approach to optimal configuration of stream processing systems. In: 2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS). pp. 39–48. IEEE (2016)

16. Kasurinen, J., Knutas, A.: Publication trends in gamification: A systematic mapping study. *Computer Science Review* **27**, 33–44 (2018)
17. Lieder, F., Griffiths, T.L.: Helping people make better decisions using optimal gamification. In: *Proceedings of the 38th Annual Conference of the Cognitive Science Society*. pp. 2075–2080 (2016)
18. Martin, J., Torres, D., Fernandez, A., Pravisani, S., Briend, G.: Using Citizen Science Gamification in Agriculture Collaborative Knowledge Production. In: *Proceedings of the XIX International Conference on Human Computer Interaction*. p. 8 (2018)
19. McKay, M.D., Beckman, R.J., Conover, W.J.: Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **21**(2), 239–245 (1979)
20. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. *Nature* **518**(7540), 529–533 (2015). <https://doi.org/10.1038/nature14236>, <https://doi.org/10.1038/nature14236>
21. Özdener, N.: Gamification for enhancing web 2.0 based educational activities: The case of pre-service grade school teachers using educational wiki pages. *Telematics and Informatics* **35**(3), 564–578 (2018)
22. Padhy, N., Panigrahi, R.: Multi relational data mining approaches: A data mining technique. *arXiv preprint arXiv:1211.3871* (2012)
23. Radoff, J.: *Game on: Energize your business with social media games*. John Wiley & Sons (2011)
24. Rasmussen, C.E., Nickisch, H.: Gaussian processes for machine learning (gpml) toolbox. *Journal of machine learning research* **11**(Nov), 3011–3015 (2010)
25. Seaborn, K., Fels, D.I.: Gamification in theory and action: A survey. *International Journal of Human-Computer Studies* **74**, 14–31 (2015)
26. Shahriari, B., Swersky, K., Wang, Z., Adams, R.P., De Freitas, N.: Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE* **104**(1), 148–175 (2015)
27. Sutton, R.S.: Learning to predict by the methods of temporal differences. *Machine learning* **3**(1), 9–44 (1988)
28. Sutton, R.S., Barto, A.G.: *Reinforcement learning : an introduction*. MIT Press (1998), <https://dl.acm.org/citation.cfm?id=551283>
29. Sweetser, P., Wyeth, P.: Gameflow: A model for evaluating player enjoyment in games. *Comput. Entertain.* **3**(3), 3 (Jul 2005). <https://doi.org/10.1145/1077246.1077253>, <https://doi.org/10.1145/1077246.1077253>
30. Trotter, M., Liu, G., Wood, T.: Into the storm: Descrying optimal configurations using genetic algorithms and bayesian optimization. In: *2017 IEEE 2nd International Workshops on Foundations and Applications of Self\* Systems (FAS\* W)*. pp. 175–180. IEEE (2017)
31. Trotter, M., Wood, T., Hwang, J.: Forecasting a storm: Divining optimal configurations using genetic algorithms and supervised learning. In: *2019 IEEE International Conference on Autonomic Computing (ICAC)*. pp. 136–146. IEEE (2019)
32. Vaquero, L.M., Cuadrado, F.: Auto-tuning distributed stream processing systems using reinforcement learning. *arXiv preprint arXiv:1809.05495* (2018)