Story Creation Algorithm Using Q-Learning in a 2D Action RPG Video Game

Diego Fernández-Samillán
Universidad Peruana de Ciencias Aplicadas (UPC)
Lima, Perú
u201310928@upc.edu.pe

Carlos Guizado-Díaz
Universidad Peruana de Ciencias Aplicadas (UPC)
Lima, Perú
u201312165@upc.edu.pe

Willy Ugarte
Universidad Peruana de Ciencias Aplicadas (UPC)
Lima, Perú
willy.ugarte@upc.pe

***Abstract*—In this paper, we try to solve the problem of making video games with multiple storylines. To do this, we developed an algorithm capable of generating variations in the game's story through altering the behaviors of the characters. For this task, we use Q-Learning. Our results indicate that this algorithm may be viable to be used in video games for commercial purposes.**

I. Introduction

The video game industry has grown exponentially in the last decade, generating more income than the film and music industry combined (from EconoTimes - "The Gaming Industry Is Now Bigger Than Hollywood" - https://bit.ly/36SPDXj). This also means that the industry has become even more competitive than before. With hundreds of video games published daily (Gaming Shift - "How Many Video Games Exist?" - https://bit.ly/3flkyPD), developers are always looking for ways to stand out from the competition.

A rather striking way is to develop video games with multiple story-lines, where the decisions of the players affect the game world. Thus, in addition to providing greater immersion to the player, they increase game's re-playability.

However, programming each variant of the story, both the dialogues and the behavior of the characters, among other things, considerably increases costs and production times. Solving this, many developers have tried to create an algorithm capable of procedural generating the narrative of a video game.

Despite the fact that there are already methods to procedural generate different parts of a video game, such as the settings, the characters, and even the graphics and music, generating the narrative is a much bigger problem since, unlike the other components that can be procedural generated individually, the narrative of a video game includes the rest of the components, which makes it more complex.

Furthermore, a narrative does not follow a pattern as clear as a setting or a character, and the number of variations it can have is exponentially greater, making its procedural generation an interesting challenge [1]. In addition, to procedural generate video game content, an element is usually chosen as a cell.

This element represents a basic atomic unit of the content to be generated (as in the case of music generation, the cell can be a musical note). Then, a series of logical rules are written that determine how cells should be combined and which combinations are better and worse.

Thus, through the combination of these elements, more complex things are formed (i.e., from notes they form melodies, from fractals they form graphics, etc.). However, in the case of story generation, the cell element is not entirely clear, which makes things even more difficult.

On the other hand, some more sophisticated solutions use a dataset to find the logical rules using machine learning methods (i.e., in the case of melody generation, methods such as deep learning can find, from a dataset of melodies, which combinations of music notes sound better or worse [2], [3]).

However, in the case of story generation, there is no dataset of video game stories, so all these methods are not feasible. Because of that, many developers have chosen to program logic rules and heuristic functions.

However, there are machine learning methods that do not need a formal dataset, such as reinforcement learning methods, which mostly require only one agent, one environment, that the agent can execute actions on the environment and that this responds with a number that represents a reward and with a new status of the environment.

These reinforcement learning methods adapt well to our problem not only due to the fact that they do not require a dataset, but also due to the fact that they adapt easily to the architecture of a video game, since they always naturally have a series of agents that execute actions in an environment, changing their state continuously.

The structure that we will follow in this paper is: first, the Background/Context, where we will describe the theoretical framework that encompasses this problem and our solution; second, the Main Contribution, where we will describe our solution to the problem; third, the Related Works, where we will make a brief state of the art of the best current solutions and indicate how our solution differs from the others; fourth, the Experiments, where we will carry out tests to verify the validity of our solution; and finally the Conclusions, where we will explain the final conclusions of our work and say some recommendations for future works.

ISSN 2305-7254

II. Background/Context

First some context, it happens that video game stories differ from literature and film stories not only in that they must be interactive (because the player is part of the story and it progresses with him), but also in many other forms. Given the unique characteristics of video games, and for the reader's understanding, we will define the components that make up the story and narrative of a video game.

According to [1], a Video game must contain:

- **Narrative:** A concept that encompasses the game's history and discourse (i.e., In the Super Mario Bros video game, the narrative represents all the experiences that the player lives, from the beginning to the end of the game [4]).
- **History:** The events that occur to the characters in the narrative, this include plot and space (i.e., In the Super Mario Bros video game, the story consists of the background story, the princess being kidnapped, and the story the player actually plays, Mario going through the 8 worlds to rescue the princess, facing enemies and going through obstacles [4]).
- **Speech:** The way the story is presented to the player, and can be through cut scenes, text boxes, or simply through the gameplay (i.e., In the Super Mario Bros video game, the story is shown to the player through gameplay, there are no cut scenes or text boxes [4]).
- **Space:** The characters, places, objects, and all the elements found in the fictional world of the game, be they tangible or abstract objects (i.e., In the Super Mario Bros video game, the space is the levels and their elements, the power-ups and the characters and enemies [4]).
- **Plot:** A series of events that occur to the characters, with a structure that orders the events based on time, and the events that occur have a cause-effect relationship between them (i.e., In the Super Mario Bros video game, the plot is the kidnapping of the princess, the order in which Mario must cross the levels and their obstacles, in addition to the end of the game, in which you save the princess [4]).

To procedural generate a video game narrative, most, if not all of the studios, have focused on automating the game's story, while few have focused on the speech [1]. Usually, they use some translator module to generate the game's speech.

This is because, when generating the game story, meaning or semantics are also generated, so a module that transforms data into natural language is more than enough to generate speech, perhaps not of the best quality, but until someone finds an efficient method of procedural generating a story that is credible, consistent, and emotionally meaningful to the player, there is little point in generating speech.

There are some methods to procedural generate the story of a video game using stories archetypes. A story archetype is a general structure that describes the components that make up a story.

There are various archetypes, such as the popular hero's journey, among others [5]. However, we believe that there is no need to use an archetype to procedural generate a story.

This is because, since every story can be built through the interactions of its characters, we believe that this is enough to generate more complex structures. To carry out our proposal, we use Q-Learning, which is a reinforcement learning algorithm.

Reinforcement learning is a learning paradigm within the machine learning area, which is based on programming intelligent agents using a system of rewards and punishments, without the need to specify how the problems should be solved.

Agents are executing actions in a dynamic environment, looking for ways to maximize the reward and minimize the punishment. For this, a reward function is usually used, which estimates how much reward the agent will get if he performs certain actions. Also, the agents receive a new state of the environment each time they execute an action.

There are many variations of reinforcement learning, some trying to optimize the time the agent searches for new solutions and the time it seeks to exploit the solutions he has already found, others that take into account long-term rewards, where he may have to start with little reward to get the maximum reward at the end [6].

After analyzing the different reinforcement learning methods [7], we decided to use Q-Learning [8], this because it does not require an environment model, but rather is based on the agent model and a Q function that predicts the reward that the agent will obtain.

Basically, the agent chooses from time to time an action to take given the reward that will be immediately granted and the expected future reward, which depends on a Q-function and a Q-table (a table where the function saves the expected rewards given an action and a state).

After executing the action, it observes the reward and modify the value in the Q-table using the Bellman's equation, so that it adapts and can better predict the rewards that are obtained by taking certain actions.

The simplicity of this algorithm adapts very well to a video game, since we do not make a model of the environment (this is simply a scenario with obstacles), but we do make a complex model of the characters, which would be the agents in this case, and they are the ones who execute actions and expect a reward.

III. Main Contribution

There are multiple approaches to automate the story of a game, most try to automate the generation of the plot [9]–[13], others try to automate the generation of the space [14], [15], while others try to automate both [16]–[19].

The degree of automation is extremely variable. For example, in MEXICA [10], the authors try to stick together a number of plot fragments, which are constrained by a set of pre and post-conditions, while J. Harris and R. Young [19] develop a system which predicts when the player is going to perform an action that invalidates the plot, and modify the

space in advance in order to make the action impossible to attempt.

We have a different proposal, in which we use Q-Learning to generate the story. With our proposal, we seek not only to generate the story of the game, but also to reduce the work for developers by reducing the amount of heuristic rules to be implemented in the game.

To begin with, we think that there is no need for story archetypes, because the story is generated naturally when various intelligent agents interact with each other. For this reason, we propose to equip the characters with an artificial intelligence capable of making decisions that affect the game world, to build the story based on the actions of the characters.

First, we describe the space of the game. To do this, we create a character template, which will have stats that determine his abilities, his motivations and his personality (our solution implements the 16 personality types, based on the study by Myers Briggs - 16Personalities - https://bit.ly/2IW92hE).

The character template will also have an inventory (so we also have to describe all the possible objects of the game) and a relationship tracker (which tracks the relation this character have with the other character, i.e., family relationship, friendship, alliance, hostility).

Seen from a programming point of view, space can be understood as a set of variables that represent the elements of the game, while the plot can be understood as functions or methods that alter the value of those variables.

So, the next step is to create a set of functions that will represent the possible actions in the game. Until now, the algorithm is a fairly common implementation, you describe the space and then you describe the combination rules that will form the plot.

However, below we do something that will significantly decrease the implementation time and the workload. It turns out that, for commercial games, the space is very large, and the quantity of rules are even greater.

The number of rules that should be implemented is outrageous, and even worse, getting these rules to be of high quality so that each behavior makes sense is an even more complicated task.

So, the solution would be to implement a version of Q-Learning. Each character will be an agent, and the possible actions to be performed will be the actions that it executes in the environment.

At the beginning of the game, and every time the character finishes an action, the character will execute a new action based on a Q-function that tells him which action is the one that will give him the best reward given his current state. To determine the expected reward, the Q function make a prediction based on the present values of the character stats and the expected values.

Then, when the character completes the action, it will obtain information about the success or failure of that action, based on whether their stats improved or get worse (i.e., the energy stat is represented by a floating variable, the amount of energy expected to be spent is taken into account and compared to the amount of energy that the character actually spent, thus, it is known if the result was satisfactory or not).

In this way, using the values of the stats, we can build a reward function for our Q-function that predicts rewards and also adjust it to adapt to reality (a diagram of this can be seen on Fig. 1).

So, we implement the agent model (characters with their stats, interests, inventory and human relations), then we implement the possible actions to be performed (functions that make the characters execute actions in the environment), and then we implemented our Q-function so that the behavior of the characters makes sense.

In this way, we no longer have to spend time or effort implementing logical rules for all possible cases of the behavior of each character, just as we no longer have to worry about the quality of these rules, since their quality will improve as the agent is learning. This optimization has the potential to dramatically reduce the amount of work and amount of hours required to develop a video game with procedural generated stories.

IV. Experiments

The method used to validate that our solution reduces the amount of work is quite simple. First, we implemented two versions of the same game, one following the classical method based purely on heuristics and rules, and ours, using Q-Learning to reduce the number of them.

Then, in order to verify that the method based on Q-Learning actually reduced the work time and workload, we gathered a group of 15 programmers, where 5 were beginners on the field of video game development (less than 2 years on the field), 5 were were intermediate (2 to 5 years on the field) and 5 were experts (more than 5 years on the field).

We asked each one of them to review all the necessary functions to implement both the Q-Learning-based method and the traditional method (based on heuristics), and to rate each function with a number from 1 to 10 (1 is less and 10 is more) indicating how long it would take to implement this function and how much workload it has. Although the method based on Q-Learning had fewer functions, they took a little more time and work to implement, so doing this test was necessary.

Given that the functions were divided into 2 groups (Q-Learning group and group based on heuristics), after performing these tests the time and workload values of each function were added for each group, in order to obtain the result of how much time and how much workload would it take for each person to implement not a single function, but all the functions of each group.

An average of these results was obtained for each group of 5 people (beginners, intermediate and experts). These results are shown in Fig. 2a and 2b.

In addition, we include a third row of results in Figures 2a and 2b, where it is indicated how much the time reduction and workload reduction will be if the Q-Learning method is used instead of the heuristic-based method. The results show that

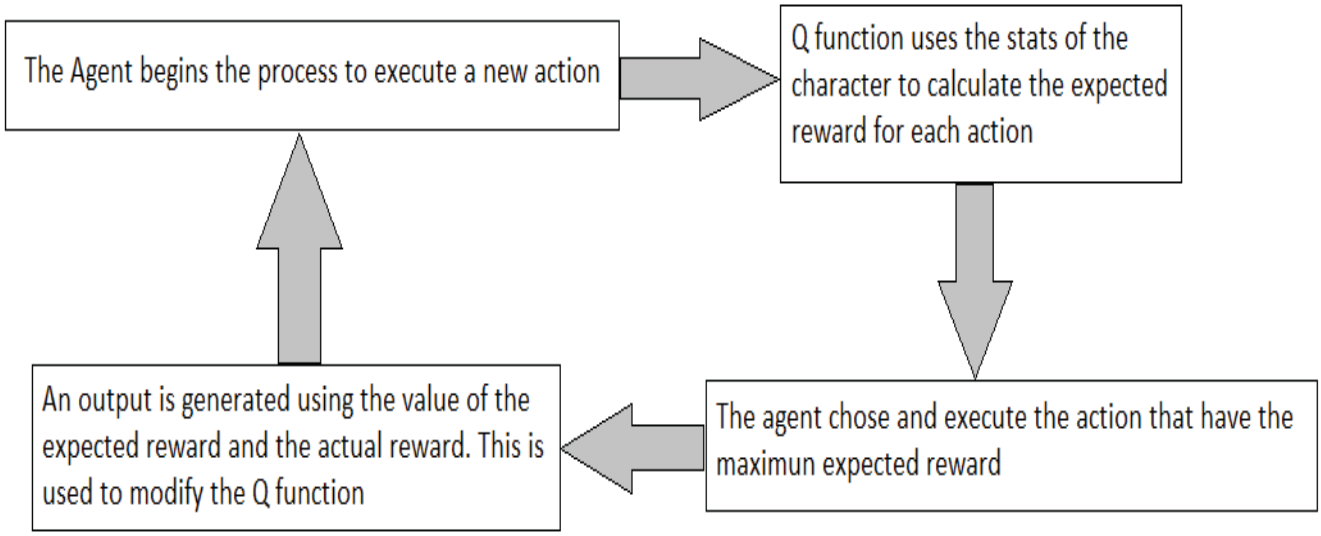


Fig. 1. The general process of the algorithm

Method	Beginner	Intermediate	Advanced
Q-Learning Method	28.4	26.4	27.4
Heuristic-Based Method	70.0	64.4	74.2
Time Reduction	59.4%	59.0%	63.1%

(a) Time

Method	Beginner	Intermediate	Advanced
Q-Learning Method	29.4	27.2	28.6
Heuristic-Based Method	71.2	71.0	78.6
Work Reduction	58.7%	61.7%	63.6%

(b) Workload

Fig. 2. Effort Tests

TABLE I. ANSWERS TO QUESTIONS 1 TO 5

Test subject ID	Q1	Q2	Q3	Q4	Q5
1	4	4	2	1	5
2	3	4	3	2	4
3	3	3	2	2	4
4	4	3	1	1	3
5	3	4	2	1	3
6	4	4	3	2	4
7	3	4	2	1	5
8	3	3	2	2	4
9	4	3	3	2	3
10	4	2	2	1	5
Average	3.5	3.4	2.2	1.5	4.0

the method based on Q-Learning significantly reduces development time and workload compared to traditional methods based on heuristics.

However, it is also necessary to compare these 2 methods in terms of similarity, to verify if the method based on Q-Learning produces the same results as traditional methods based on heuristics. For this, in Fig. 3a, 15 test cases were raised, where an NPC with certain stats was subjected to a stimulus, to which he responded using each of the 2 methods.

Then we compared the 2 reactions for each case (see Figure 3b), and we checked how many cases of similarity there were. From the 15 cases, there was a match in 11 of them, which means a similarity of 73.33% between the two methods.

The error is probably due to the fact that during training the reward function was not the best pick. However, these differences are minor, and what really matters is whether the player is capable of differentiating both versions of the game. Also, we need to validate that the generated stories make sense and that the game really feels procedural generated.

Video games have a high subjective load, which makes it difficult to validate the elements of the game. Because of that, the Game User Research (GUR) area was developed, an area created to develop methods to verify if a video game meets the desired requirements [20].

Within this area, there are various techniques, of which we will use a playtest to validate that the stories generated by the video game have coherence, are entertaining and are different

Variable	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15
Health	0.8	0.4	0.4	0.4	1.0	0.8	0.5	0.1	0.9	0.3	0.9	0.4	0.5	0.4	0.9
Stamina	0.8	0.3	0.2	0.2	0.2	0.5	1.0	0.6	0.3	0.5	0.2	0.6	1.0	0.6	0.3
Hunger	0.7	0.8	0.3	0.3	0.2	0.4	0.2	0.7	0.5	0.2	0.9	0.2	0.2	0.3	0.5
Strength	0.7	0.1	0.2	0.2	0.2	0.3	0.7	0.9	0.7	0.1	0.5	0.1	0.7	0.1	0.7
Dexterity	0.7	0.3	0.7	0.7	0.7	0.4	0.1	0.2	0.1	0.4	0.9	0.3	0.1	0.3	0.1
Constitution	0.5	0.7	0.5	0.5	0.5	0.1	0.6	0.6	0.1	0.1	1.0	0.2	0.6	0.2	0.1
Intelligence	0.4	0.2	0.4	0.4	0.4	0.3	0.6	1.0	0.1	0.4	0.8	0.6	0.6	0.1	0.1
Wisdom	0.4	0.2	0.4	0.4	0.2	0.5	0.7	0.6	1.0	0.5	0.4	1.0	0.7	0.2	1.0
Charisma	0.4	0.2	0.4	0.4	0.3	0.2	0.8	0.2	0.8	0.8	0.8	0.1	0.8	0.1	0.8
Food in inventory	0	25	0	0	0	5	0	50	0	0	0	0	0	0	0
Rock in inventory	0	0	25	25	5	5	0	50	15	0	0	30	0	0	15
Stimulus	None	Very Hungry	Attack from NPC Type B	Low Health	None	Attack from NPC Type A	None	None	None	Attack from NPC Type C	Very Hungry	Low Health	Attack from NPC Type A	Low Health	Attack from NPC Type B
Reaction (Q-Learning)	Search Food	Eat	Escape	Rest	Take a walk	Attack	Rest	Take a walk	Search Food	Attack	Search Food	Rest	Attack	Take a walk	Attack
Reaction (Heuristic Based Method)	Search Food	Eat	Attack	Rest	Take a walk	Attack	Rest	Rest	Search Food	Escape	Search Food	Rest	Attack	Rest	Attack

(a) Cases Results

NPC Type	Health	Stamina	Hunger	Strength	Dexterity	Constitution	Intelligence	Wisdom	Charisma	Food in inventory	Rock in inventory
A	0.8	0.7	0.2	0.9	0.7	0.7	0.4	0.5	0.2	0	0
B	0.2	0.3	0.8	0.1	0.3	0.3	0.7	0.5	0.8	5	5
C	1.0	0.7	0.2	0.9	0.7	0.8	0.1	0.5	0.2	0	0

(b) NPC Tests

Fig. 3. Similarity Tests

from each other. A playtest will be carried out on 10 people from our target audience (see Table I), which is made up of people between 15 and 27 years old, who mainly like games where the story is modified based on their actions (i.e., Stanley Parable [21]).

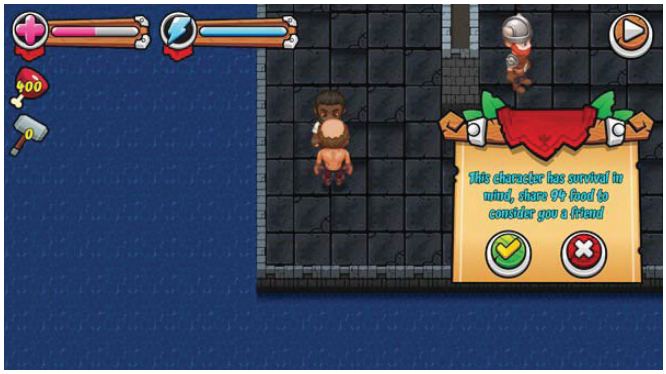
For this purpose, we developed a video game (see Fig. 4) to experiment with a duration around 5 to 10 minutes, so playing them 3 times shouldn't take even an hour. After, we carry out experiments with the following instructions:

1) These people must record the video game and their faces while playing at least 3 times.
2) They will first play the version of the game that use the classical approach two times, then, they will play the version of the game that uses our approach, but they won't know that.
3) Then, they will be asked a questionnaire with the following questions:

Q1: Were the events in the game coherent?
Q2: Did you feel that your decisions affect the game world?
Q3: Did you feel changes in the game world when you played the video game again?
Q4: Did you want to make a decision and could not because of the limitation of the actions?
Q5: Did you feel that you were playing the same game the 3 times?

Questions 1 to 5 must be answered with a number from 1 (Totally Disagree) and 5 (Totally Agree).

Table I shows the responses of the 10 people (see https://tinyurl.com/y3mexxju).



(a) Screenshot 1

(b) Screenshot 2

Fig. 4. Screenshots of our proposal of a Video game

V. Related Works

At this point, we will review and compare our solution with the current state of the art. The engine Creation Of Novel Adventure Narrative (CONAN) tries to create a story through the procedural creation of quest.

To do this, it describes all the elements of the game (characters and their preferences and objectives, places and actions) and describes a series of predicates that indicate relationships of belonging, possession or any other relationships between the elements of the game.

It also describes a series of logical rules that indicate the parameters that an action receives, the prerequisites for executing said action and the consequences of executing it. Afterwards, each character will search for the shortest route to reach their objective, where the paths to reach this are the actions, and the weights of each action are determined by the preferences of the characters.

The complexity of the quests will be determined by the quality of the initial mapping [22]. Unlike this algorithm, ours uses reinforcement learning so that each character forges their personality as the game progresses, reducing the amount of work.

There are also jobs like Tale-Spin, which describes characters in space with goals and personalities, and then uses a series of logical rules so that the characters can try to achieve their goals [23]. There are many similar works, which use well-defined logical or grammatical rules to perform combinations, and despite the fact that it works, manual work can be quite considerable.

There have also been works that try to generate the story using crowd-sourcing methods, where a high number of anonymous authors offer story snippets, and an algorithm is in charge of combining these snippets to produce the final story.

However, these methods often have consistency problems [24], or writers are asked for several restrictions, restrictions that make work more difficult by depending on human writers [25].

Other methods include creating fictional gadgets that are added to the story generator. In case the generator encounters any problems for characters to achieve their goals, gadgets will use analogy-based reasoning to find alternate routes for the character [26].

Although it is an interesting solution to correct certain problems, it can be seen as a simple extension of the logical rules of the system, so it is still not a sufficiently automatic process.

Finally, the last form of story automation for a game is found in the video game “Slaves to Armok: God of Blood, Chapter II: Dwarf Fortress”, commonly known as “Dwarf Fortress”.

In this game, a world is first procedural generated, using a noise algorithm and a series of logical rules. Then, places and characters are procedural generated following a similar method.

Then, the game simulates a process of historical evolution, since it simulates a number of years (the user can set the quantity of years, but normally it is several hundred years, up to thousands of years).

During this process of historical evolution, each character makes his own decisions, which affects the world around him and forms the lore of the game. For this process to work, it is necessary to describe all the elements of the game world and write a large number of logical rules.

In addition, each element of the game world stores information about the rest of the elements with which it interacts. The problem with Dwarf Fortress is that, in order to get to an impressive level of detail, they program an impressive number of elements and their combination rules [27].

The difference between the algorithm of this game and ours is not so much in the results, but in the development time, since our method can produce the same results but in less time and with a lower workload for the programmers.

Therefore, our solution does contribute to improving the current state of the art, however, it is necessary to carry out tests to validate our proposal.

VI. Conclusion

After reviewing the results, it is clear that the method based on Q-Learning reduces significantly the development time and workload if we compare it to the traditional method based on rules and heuristics.

In addition, it seems that the more experience you have in the field of video game development, the reduction of development time and workload will be greater, although we need to do tests on a larger scale to validate the latter. On the other hand, the similarity tests showed that the method based on Q-Learning does not always produce the same results.

Even so, the tests made to the users through the Game User Research show that the users found the responses of the NPCs coherent, so that, despite the fact that the method based on Q-Learning produces different responses to the traditional methods, they continue being valid and coherent answers.

In any case, it is possible that by improving the reward function the method based on Q-Learning could produce the same results as traditional methods, although future research would be necessary to validate this. Another thing to keep in mind is that users didn't feel many variations in the game when they played it again.

The latter may be due to the fact that, due to time constraints, the game we implemented was small, and since the initial state of the game space was so small, the number of combinations was not very large. In any case, to confirm or disprove this, it would be necessary to use the method based on Q-Learning in a much larger-scale video game.

Either way, while the results produced by the Q-Learning method are not exactly the same as those produced by traditional methods based on rules and heuristics, they are results similar enough to be valid, and most importantly, it manages to successfully reduce development time and workload, so it is viable to be used in video games for commercial purposes.

Finally, our recommendation for future works would be to concentrate on finding a way to create a dataset that contains different video game stories, and for each one, it must have a description of all the elements that make up the game and their respective parameters.

In addition, the dataset must contain descriptions of all the relationships that exist between the elements of the game, both human relationships (such as friendship or family), relationships of belonging (this object belongs to such person or such tavern is found in such town), logical relationships (if you attack someone, they will get mad at you) and all kinds of relationships necessary to fully describe the story of the game.

This is our recommendation because, if these datasets existed, other machine learning methods could be applied to solve this problem, which could lead to new methods with better results and less work time, similar to Music Sheet Classification [28] or soft constraints [29].

References

[1] B. A. Kybartas and R. Bidarra, "A survey on story generation techniques for authoring computational narratives," *IEEE Trans. Comput. Intell. AI Games*, vol. 9, no. 3, pp. 239–253, 2017.

[2] J. Briot, G. Hadjeres, and F. Pachet, *Deep Learning Techniques for Music Generation*. Springer, 2020.

[3] J. Briot and F. Pachet, "Deep learning for music generation: challenges and directions," *Neural Comput. Appl.*, vol. 32, no. 4, 2020.

[4] S. Miyamoto, H. Yamauchi, and T. Tezuka, "Super mario bros," *Nintendo Entertainment System. Nintendo*, 1985.

[5] J. Campbell, *The hero with a thousand faces*. New World Library, 2008, vol. 17.

[6] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.

[7] C. Sammut and G. I. Webb, Eds., *Encyclopedia of Machine Learning and Data Mining*. Springer, 2017.

[8] M. A. S. Santibanez, "Building an artificial cerebellum using a system of distributed q-learning agents," Ph.D. dissertation, University of Arizona, USA, 2010.

[9] S. W. Haas, "The creative process: A computer model of storytelling and creativity, by scott r. turner," *J. Am. Soc. Inf. Sci.*, vol. 47, no. 3, pp. 252–254, 1996.

[10] R. P. y Pérez and M. Sharples, "MEXICA: A computer model of a cognitive account of creative writing," *J. Exp. Theor. Artif. Intell.*, vol. 13, no. 2, pp. 119–139, 2001.

[11] T. Ong and J. J. Leggett, "A genetic algorithm approach to interactive narrative generation," in *Hypertext*. ACM, 2004, pp. 181–182.

[12] E. Clark, A. S. Ross, C. Tan, Y. Ji, and N. A. Smith, "Creative writing with a machine in the loop: Case studies on slogans and stories," in *IUI*. ACM, 2018, pp. 329–340.

[13] M. Roemmele and A. S. Gordon, "Automated assistance for creative writing with an RNN language model," in *IUI Companion*. ACM, 2018, pp. 21:1–21:2.

[14] D. Delgado, J. Magalhães, and N. Correia, "Assisted news reading with automated illustration," in *ACM Multimedia*. ACM, 2010, pp. 1647–1650.

[15] K. Schwarz, P. Rojtberg, J. Caspar, I. Gurevych, M. Goesele, and H. P. A. Lensch, "Text-to-video: Story illustration from online photo collections," in *KES (4)*, ser. Lecture Notes in Computer Science, vol. 6279. Springer, 2010, pp. 402–409.

[16] I. Swartjes, E. Kruizinga, and M. Theune, "Let's pretend I had a sword: Late commitment in emergent narrative," in *ICIDS*, ser. Lecture Notes in Computer Science, vol. 5334. Springer, 2008, pp. 264–267.

[17] I. Swartjes and M. Theune, "A fabula model for emergent narrative," in *TIDSE*, ser. Lecture Notes in Computer Science, vol. 4326. Springer, 2006, pp. 49–60.

[18] M. O. Riedl, C. J. Saretto, and R. M. Young, "Managing interaction between users and agents in a multi-agent storytelling environment," in *AAMAS*. ACM, 2003, pp. 741–748.

[19] J. Harris and R. M. Young, "Proactive mediation in plan-based narrative environments," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 3, pp. 233–244, 2009.

[20] P. Mirza-Babaei, V. Zammitto, J. Niesenhaus, M. Sangin, and L. E. Nacke, "Games user research: practice, methods, and applications," in *CHI Extended Abstracts*. ACM, 2013, pp. 3219–3222.

[21] D. Wreden and W. Pugh, "The stanley parable," *Galactic Cafe*, 2013.

[22] V. Breault, S. Ouellet, and J. Davies, "Let CONAN tell you a story: Procedural quest generation," *CoRR*, vol. abs/1808.06217, 2018.

[23] J. R. Meehan, "Tale-spin, an interactive program that writes stories," in *IJCAI*. William Kaufmann, 1977, pp. 91–98.

[24] R. Swanson and A. S. Gordon, "Say anything: A massively collaborative open domain story writing companion," in *ICIDS*, ser. Lecture Notes in Computer Science, vol. 5334. Springer, 2008, pp. 32–40.

[25] B. Li, S. Lee-Urban, and M. Riedl, "Crowdsourcing interactive fiction games," in *FDG*. Society for the Advancement of the Science of Digital Games, 2013, pp. 431–432.

[26] B. Li and M. O. Riedl, "A phone that cures your flu: Generating imaginary gadgets in fictions with planning and analogies," in *Intelligent Narrative Technologies*, ser. AAAI Workshops. AAAI, 2011.

[27] T. Adams and Z. Adams, "Slaves to armok: God of blood chapter II: Dwarf fortress," *PC Game. Bay*, vol. 12, 2006.

[28] D. J. Lozano-Mejía, E. P. Vega-Uribe, and W. Ugarte, "Content-based image classification for sheet music books recognition," in *2020 IEEE Engineering International Research Conference (EIRCON)*, 2020.

[29] W. Ugarte, P. Boizumault, S. Loudni, B. Crémilleux, and A. Lepailleur, "Soft constraints for pattern mining," *J. Intell. Inf. Syst.*, vol. 44, no. 2, pp. 193–221, 2015.