Proceedings of the 2023 International Conference on Machine Learning and Automation
DOI: 10.54254/2755-2721/38/20230550

Application of machine learning algorithms to character interaction in single-player role-playing games

Haoran Huang

Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, China, 210023

836034509@qq.com

Abstract. Character interaction in single-player role-playing games refers to the role interactions between the player character and the non-player characters (NPCs) in the game including dialogue, social interaction, cooperation, and competition, which are designed and implemented to achieve certain game goals and effects. This paper summarizes and compares machine learning algorithms applied to game development using a paper review and theoretical analysis, aiming to provide new solutions for optimizing and improving algorithms commonly used in game design. The ADEM algorithm can automate the evaluation of semantic relevance and logical coherence of character dialogues, but it is more costly in the pre-training phase and more difficult to perform domain adaptation for specific game content. The BERT algorithm can improve the speed of character dialogue generation through the training of unlabeled text, and then improve the flexibility of the dialogue through a small amount of labeled text for domain adaptation and fine-tuning, but it lacks in the generation of smooth and complex text and other aspects. The GAN algorithm can learn from a single or a small number of action sequences to achieve the transformation and mixing of character actions, thus generating multiple action gestures. However, its training process may suffer from problems such as pattern collapse leading to degradation in the quality of the generation, and it is also difficult to set a uniform objective criterion to determine the loss function.

Keywords: Machine Learning, Role Playing Game (RPG), ADEM, BRET, GAN.

1. Introduction

The single-player role-playing game is a type of game that allows players to take on the role of one or more characters and improve their characters' abilities and experiences by completing quests, exploring the world, and so on. Single-player role-playing games usually have rich plots, varied choices, and free play, allowing players to shape their characters and stories to their liking. Character interaction in single-player role-playing games refers to the role interaction between the player character and the non-player characters (NPCs) in the game, including dialogue, social interaction, cooperation, competition, etc., through the design and implementation of the role interaction to achieve certain game objectives and effects [1]. Based on the application of machine learning algorithms, the development of games has also been positively affected and promoted by continuous breakthroughs in machine learning. Therefore, it is necessary to summarize and discuss the commonly used machine learning algorithms for single-player

DOI: 10.54254/2755-2721/38/20230550

game character interaction development and design, to better increase the immersion and fun of the game.

This paper uses a paper review and theoretical analysis to summarize and compare machine learning algorithms applied to game development, and provide new solutions for improving the optimization of algorithms commonly used in game design. Comparative analysis of the algorithms can help to apply the developed machine learning results more widely to the gaming domain, thus improving the intelligence of game characters, e.g., by adapting or mimicking based on players' behaviors and preferences. The increased intelligence of game characters can create more diverse and personalized game characters, allowing players to encounter different situations and choices in the game, enriching the content and experience of the game [2].

2. Characteristics of character interaction in single-player role-playing games

Different genres and styles of single-player role-playing games may have different character interaction logics and characteristics, and factors that usually need to be taken into account include the genre of the game, the setting and personality of the characters, and the form of character interaction.

2.1. The genre of the game

It is possible to classify RPG games into different genres based on their style and gameplay. Depending on the style and theme of the game, they can be categorized as Western Role-Playing Game (WRPG), Japanese Role-Playing Game (JRPG), and Chinese Role-Playing Game (CRPG). [3].

WRPGs place more emphasis on freedom, choice and consequence, open world, and sandbox gameplay. allowing players to choose their character's appearance, gender, race, profession, skills, etc. according to their preferences and styles, and to influence the state of the game world or events through interactions with NPCs. JRPGs place more emphasis on plot, characters, turn-based combat, fixed routes, etc., so that players can immerse themselves in an interesting and challenging story, and add to the game's emotional level and character CRPGs usually place more emphasis on culture, history, etc. so that players can experience a simulated environment close to the real world or historical era, and feel the charm and value of various cultures through interaction with NPCs.

Based on gameplay and mechanics, games can be categorized into (Turn-based Role-playing Game (TRPG), Action Role-playing Game (ARPG) and Strategy Role-playing Game (SRPG). These types mainly reflect different battle modes and strategy requirements. Among them, TRPG focuses on thinking and planning, allowing players to choose appropriate actions and skills in each turn. Through communication with NPCs, they learn about the story's background, obtain quests or information, and even change the direction of the game's plot by influencing NPCs' attitudes or relationships. While ARPG focuses on action and reaction. By using different weapons, skills, and strategies to fight with NPCs in real-time battles, players can experience the action and strategy of the game. SRPGs focus on management and command, where players can control multiple characters or units on a large map to fight or build, trade, ally, and compete with NPCs.

2.2. The setting and personality of characters

A character's setting refers to basic attributes such as the character's appearance, gender, race, profession, and skills, as well as deeper attributes such as the character's backstory, goals, and motivations. The character's setting affects the feasibility and variety of character interactions. In addition, the character's setting also affects the ease and outcome of character interactions.

A character's personality is the psychological characteristics of a character's thoughts, feelings, attitudes, and values. The character's personality affects the atmosphere and affects character interaction. At the same time, the character's personality also affects the choices and consequences of character interactions.

DOI: 10.54254/2755-2721/38/20230550

2.3. Forms of Character Interaction

According to different game types and design purposes, haracter interactions in single-player role-playing games can take four forms: dialogue interactions, combat interactions, trading interactions, and social interactions. [4].

Dialog Interaction: players can communicate with NPCs by choosing different dialog options. The main purpose of this method is to help players understand the story background, obtain quests or information, influence the attitude or relationship of NPCs, and even change the plot direction of the game.

Battle Interaction: Players experience the action and strategy of the game through battles with NPCs. Battle interaction can be categorized into two modes: real-time battle and turn-based battle.

Trading Interaction: Players exchange items or currency with NPCs to increase the complexity and interest of the game's economic system and item system.

Social Interaction: Players establish different levels and types of social relationships with NPCs, such as friendship, love, marriage, hostility, etc., to increase the emotional level and characterization of the game.

2.4. Excellent Character Interaction

Excellent character interaction, including language, action, demeanor, and the environment they are in, should be in line with the theme, style and gameplay of the game. For example, the character interaction needed in a horror survival game needs to create a tense and fearful atmosphere in the game, so that players can feel the pressure of scarce resources, crises and life and death decisions. In addition, game interactions should be in line with the character's setting and behavioral logic, and show his or her personality and characteristics as much as possible. For multiple interactions, the content should also be in line with the internal logic of character growth and plot development. At the same time, there should be enough diversity and flexibility in both content and form to provide players with different scenarios and choices to suit different types or levels of players.

3. ADEM, BERT and GAN

3.1. Automatic Dialogue Evaluation Model (ADEM)

In single-player role-playing games, the ADEM algorithm can evaluate the quality of dialog between players and NPCs, or between NPCs and NPCs. In turn, the game developer can improve the dialog strategy according to the evaluation results to achieve the purpose of improving the dialog quality.

3.1.1. Principle. The Automatic Dialogue Evaluation Model (ADEM) algorithm is a method for automatically evaluating the quality of a dialog system, proposed by Ryan Lowe et al. in 2017. Its basic idea is to utilize a new dataset of human response ratings to train a model to predict human scoring of input responses [5]. The ADEM algorithm implementation consists of two parts: a bi-directional LSTM-based response encoder, and a multi-layer perceptron-based scorer.

The bi-directional LSTM model as response encoder mainly encodes the context, reference response, and candidate response into fixed-length vector representations. The multilayer perceptron model as a scorer predicts the score of the candidate response based on the output of the response encoder [6].

In addition, the ADEM algorithm uses the mean square error as a loss function, which measures the difference between the ratings output by the rater and the human ratings, and uses stochastic gradient descent to optimize the model parameters. The ADEM scoring function is calculated as follows:

$$Score(c, r, r') = \frac{c^T M r' + r^T N r' - \alpha}{\beta}$$

3.1.2. Application in Role-Playing Games. First, the ADEM algorithm needs to load the human response score dataset and divide it into training, validation and testing sets.

DOI: 10.54254/2755-2721/38/20230550

Next, the ADEM algorithm needs to initialize the model parameters of the response encoder and the scorer, and set the hyperparameters, such as the learning rate, batch size, and number of iterations. Next, model training needs to be performed, i.e., within each iteration number, the following steps are performed:

A batch of data samples is randomly selected from the training set:

For each data sample, a response encoder was used to encode the context, reference response, and candidate response into a vector representation.

For each data sample, a rater is used to predict the ratings of the candidate responses using the output of the response encoder as input.

For each data sample, the mean square error between the ratings output by the rater and the human ratings is calculated and accumulated to obtain the batch loss.

Based on the batch loss, the gradients of the model parameters are calculated using the backpropagation algorithm and the model parameters are updated using stochastic gradient descent.

Finally, model testing is required, i.e., the correlation metrics between the model-predicted ratings and human ratings, such as Pearson correlation coefficients, Spearman rank correlation coefficients, etc., between model-predicted ratings and human ratings on the test set.

3.1.3. Result analysis. The ADEM algorithm automatically evaluates the quality of in-game dialog systems without the need for human involvement or additional labeling data. This can save cost and time in game development. In addition, the context, reference responses and candidate responses of the game dialog can be considered in a comprehensive way, not only based on lexical overlap or language model scores. This can better reflect the semantic relevance, emotional adaptation and logical coherence of the dialog, etc.

However, the ADEM algorithm requires a large-scale human scoring dataset for training, which may increase the difficulty of data collection and annotation for a multilingual, large-scale game.The scoring results of the ADEM algorithm may be affected by the quality and size of the dataset, and it may not be possible to learn a stable and generalized scoring function if there are noises, biases, or imbalances in the dataset, which this can lead to poor or unstable performance of the ADEM algorithm in some game scenarios.

In addition, ADEM algorithms may not be able to adequately take into account the specific nature of in-game dialogues, such as characters' personalities, goals, and so on. Since different languages, domains and scenarios may have different dialog characteristics and evaluation criteria, ADEM algorithms may not be able to capture these subtle or implicit differences. This makes it difficult to ensure consistency of game content across different versions.

3.2. Bidirectional Encoder Representations from Transformers (BERT)

3.2.1. Principle. Transformer is a neural network structure based on Self-Attention Mechanism, which can efficiently process sequential data such as text, speech, images, etc. Transformer encoder consists of multiple identical layers, each layer contains two sublayers: Multi-Head Self-Attention Sublayer and Feed-Forward Neural Network Sublayer. The Multi-Head Self-Attention Sublayer is used to compute the correlation between each word and other words and gives different weights to different attention heads. Feed-Forward Neural Network Sublayer is used to perform a nonlinear transformation for each word.

The input to BERT consists of the sum of three embedding vectors: word embedding, fragment embedding and position embedding. Word embeddings are based on Word Piece to process unregistered words and multilingual texts. Fragment embeddings are used to distinguish different sentences or paragraphs. Position embedding is used to represent the positional information of each element in a sequence [7].

DOI: 10.54254/2755-2721/38/20230550

3.2.2. Application in Role-Playing Games. In the Pre-training phase, a large amount of unlabeled textual data, such as Wikipedia, news, novels, etc., is used for self-supervised learning to learn common language representations.

In the domain adaptation phase, the BERT algorithm uses data from in-game dialogs for Continual Pre-training or Domain Pre-training to adapt the model to the features and knowledge of in-game dialogs.

In the Fine-tuning phase, the BERT algorithm adds an additional output layer to the last layer of the model for Fine-tuning according to the specific tasks of the in-game dialog, such as Q&A, small talk, task guidelines, etc. The Fine-tuning process is performed by providing the model with some labeled dialogue data so that the model learns how to generate appropriate responses based on the input information such as the context, role, and emotion [8].

3.2.3. Result analysis. The BERT algorithm can utilize a large amount of unlabeled text to learn the general knowledge of the language, which can improve the speed of dialogue generation. In addition, the BERT algorithm adapts to different game scenarios and characters by fine-tuning and requires only a small amount of labeled data to adjust the model parameters. This can improve the flexibility and diversity of dialog generation.

However, the BERT algorithm is a self-coding language model, which is mainly used for natural language understanding tasks rather than natural language generation tasks. As a result, it may be less suitable for generating long or complex texts, and may also lack some creativity and imagination. In addition, the BERT algorithm uses a masking mechanism in the pre-training phase to randomly mask some words in the input sentences, which may lead to some incoherent or unnatural phenomena in the model when generating dialogues, such as repeating words, semantic errors, and so on. This can affect the fluency of the dialog and the player's gaming experience, and needs to be corrected manually.

3.3. Generative Adversarial Network (GAN)

3.3.1. Principle. Generative Adversarial Network (GAN) is an unsupervised learning approach by which two neural networks play with each other. Generative Adversarial Network consists of a generative network and a discriminative network. The generative network takes random samples from the potential space as inputs, and its outputs need to mimic the real samples in the training set as much as possible. The input of the discriminative network is the real samples or the output of the generative network, and its purpose is to discriminate the output of the generative network from the real samples as much as possible. The generative network, on the other hand, has to deceive the discriminative network as much as possible. The two networks fight against each other and continuously adjust their parameters, with the ultimate goal of making the discriminative network unable to discriminate the output of the generative network from the real results [9].

The loss function of GAN is a zero-sum game, which can be defined in terms of cross entropy or least squares. The loss function reflects the adversarial relationship between the generator and the discriminator, and as the loss of one network decreases, the loss of the other network increases.

$$L(\mu_G, \mu_D) := \mathbb{E}_{x\sim\mu_{\text{ref}}, y\sim\mu_D(x)}[ln\, y] + \mathbb{E}_{x\sim\mu_{\text{G}}, y\sim\mu_D(x)}[ln(1-y)]$$

The training process of GAN can be divided into two steps:

fixing the generator G and updating the discriminator D so that D can better distinguish between real and generated samples.

fix the discriminator D and update the generator G, making G better able to deceive the discriminator D.

These two steps alternate until an equilibrium is reached where the samples generated by the generator and the real samples cannot be distinguished by the discriminator.

3.3.2. Application in Role-Playing Games. There are 5 steps required to generate the action poses of NPC using GAN [10]:

DOI: 10.54254/2755-2721/38/20230550

The first step is to prepare the training dataset, i.e., a number of images containing different npc and action gestures. These images can be obtained from existing games or other sources, or can be synthesized using other methods.

The second step is to design the network structure and parameters of the generator and discriminator. The generator usually consists of a fully connected layer and several inverse convolution layersdeconvolution layer, starting from random noise vectors, gradually increasing the size of the feature map and the number of channels, and finally outputting an image of the same size as the input image. The discriminator usually consists of several convolution layers and a fully connected layer, starting from the input image, gradually decreasing the size of the feature map and increasing the number of channels, and finally outputting a scalar value indicating the probability that the input image is real or generated. The network structure and parameters can be adapted and optimized for different tasks and datasets.

The third step is to define the loss function and optimizer. The loss function is used to measure the performance of the generator and the discriminator, usually using cross-entropy loss, which is the difference between the discriminator output and the true label. The optimizer is used to update the weights of the generator and discriminator, usually using stochastic gradient descent or its variants. The loss function and optimizer can also be tuned and optimized for different tasks and datasets.

The fourth step is to perform adversarial training. Adversarial training involves alternately updating the weights of the generator and the discriminator so that the generator can deceive the discriminator and the discriminator can recognize the generator. Adversarial training requires balancing the competitiveness between the generator and the discriminator to avoid problems such as mode collapse or oscillation. Adversarial training can be continued until the desired effect is achieved or stopping conditions are met.

The fifth step is to evaluate and test the model. Evaluating the model means using some metrics or methods to measure the performance of the model, e.g. sense perceptual loss, structural similarity index, naturalness score), etc. Testing the model means using some new random noise vectors as inputs to observe whether the model can generate npc action pose images that meet the requirements and check whether there are some defects or anomalies.

3.3.3. Result analysis. GAN can learn from a single or small number of action sequences, greatly simplifying the process of data collection and annotation.

GAN enables transformation and blending of actions, such as migrating the actions of one person to another person or animal, or fusing different action sequences into a new action sequence.

GAN can generate high-resolution and full-body action poses for various types and styles of game scenes.

The training process of GAN may be unstable and prone to mode collapse, i.e., the generator can only generate some similar or repetitive action gestures without covering the diversity of real action gestures [11].

The training process of GAN requires a lot of computational resources and time, especially when the network structure of the generator and discriminator is complex, and high performance GPUs or TPUs may be required to accelerate the training.

The training process of GAN requires appropriate hyper-parameter settings, such as learning rate, optimizer, loss function, etc. These hyper-parameters have a great impact on the performance and stability of GAN, but often there is no objective index or standard to measure the goodness of generating the action gestures, and it needs to rely on the subjective judgment of human.

4. Conclusion

By analyzing the characteristics of different types of RPG games and different forms of in-game character interactions, this paper argues that excellent character interactions, including language, actions, demeanor, and the environment in which they are located, should be in line with the game's themes,

DOI: 10.54254/2755-2721/38/20230550

styles, and gameplay, as well as the character's settings and behavioral logics, and have enough diversity and flexibility to provide players with different contexts and choices.

This paper introduces three machine learning algorithms, ADEM, BERT and GAN, which have achieved some applications in RPG games, gives their algorithmic principles and implementation processes, and analyzes their application effects:

ADEM algorithm can automatically evaluate the semantic relevance and logical coherence of character dialogues, but its cost in the pre-training stage is high, and it is more difficult to make domain adaptation for specific game contents.

The BERT algorithm can improve the speed of character dialog generation through the training of unlabeled text, and then improve the flexibility of the dialog through a small amount of labeled text for domain adaptation and fine-tuning, but it lacks in generating smooth and complex text and other aspects.

The GAN algorithm can learn from a single or a small number of action sequences to realize the transformation and mixing of character actions, thus generating multiple action gestures. However, its training process may have problems such as pattern collapse leading to a decrease in the quality of generation, and it is also difficult to set a uniform objective standard to determine the loss function.

The shortcoming of this paper lies in the fact that the results between different algorithms have not been compared through experimental methods, and thus the description of the details of the algorithm implementation may not be convincing enough.

In the future, the author will continue to learn and understand the knowledge related to machine learning, and design and develop a single-player game on his own, so as to practically apply the machine learning algorithm in the development of the game. We hope that after reading this article, readers will have a general understanding of the machine learning algorithms used in single-player role-playing games, and take this as an opportunity to further improve the machine learning algorithms to enhance the quality of the game.

References

[1] Hitchens M, Drachen A. The many faces of role-playing games[J]. International journal of role-playing, 2008 (1): 3-21.

[2] Nam S G, Ikeda K. Generation of diverse stages in turn-based role-playing game using reinforcement learning[C]//2019 IEEE Conference on Games (CoG). IEEE, 2019: 1-8.

[3] Mackay D. The fantasy role-playing game: A new performing art[M]. McFarland, 2017.

[4] Hammer J, Beltrán W, Walton J, et al. Power and control in role-playing games[M]//Role-Playing Game Studies. Routledge, 2018: 448-467.

[5] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1116–1126, Vancouver, Canada. Association for Computational Linguistics.

[6] Lowe R, Noseworthy M, Serban I V, et al. Towards an automatic turing test: Learning to evaluate dialogue responses[J]. arXiv preprint arXiv:1708.07149, 2017.

[7] Shreyashree S, Sunagar P, Rajarajeswari S, et al. A Literature Review on Bidirectional Encoder Representations from Transformers[J]. Inventive Computation and Information Technologies: Proceedings of ICICIT 2021, 2022: 305-320.

[8] Sun F, Liu J, Wu J, et al. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer[C]//Proceedings of the 28th ACM international conference on information and knowledge management. 2019: 1441-1450.

[9] Creswell A, White T, Dumoulin V, et al. Generative adversarial networks: An overview[J]. IEEE signal processing magazine, 2018, 35(1): 53-65.

[10] Jin L, Tan F, Jiang S. Generative adversarial network technologies and applications in computer vision[J]. Computational intelligence and neuroscience, 2020, 2020.

DOI: 10.54254/2755-2721/38/20230550

[11] Oliehoek F A, Savani R, Gallego-Posada J, et al. GANGs: Generative adversarial network games[J]. arXiv preprint arXiv:1712.00679, 2017.