Player Behavior Modeling for Enhancing Role-Playing Game Engagement

Sha Zhao, *Member, IEEE*, Yizhi Xu, Zhiling Luo, Jianrong Tao, *Member, IEEE*, Shijian Li, Changjie Fan, and Gang Pan, *Member, IEEE*

***Abstract*— Role-playing games (RPGs) are one of the most exciting and most rapidly expanding genres of online games. Virtual characters that are not controlled by players, have become an integral part, which helps to advance narratives of RPGs. Believable characters can enhance game engagement and further improve player retention. However, game players easily find that most characters' behaviors are limited and improbable, resulting in a less meaningful game experience. In this work, we propose a framework to model game behaviors to learn behavior patterns of human players. Based on the learned behavior patterns, it generates human-like action sequences that can be used for the design of believable virtual characters in RPGs, so as to enhance game engagement. Specifically, considering the influence of game context in behavior patterns, we integrate game context (players' levels and game classes) with actions together to model behaviors. We propose a long-term memory cell on actions and game context to learn the hidden representations. We also introduce an attention mechanism to measure the contribution of the actions previously performed to the next action. Given only one action, our model can generate action sequences by predicting the succeeding action based on the previously generated actions. The model was evaluated on a real-world data set of over 22 000 players and more than 51 million action logs of an RPG game in 21 days. The results demonstrate the state-of-the-art performance.**

***Index Terms*— Behavioral sequence generation, player behavior modeling, role-playing games (RPGs).**

I. Introduction

ROLE-PLAYING games (RPGs) are one of the most exciting and most rapidly expanding genres of online games [1]–[3], where players assume the roles of characters in a fictional setting and take responsibility for acting out these roles within a narrative, such as World of Warcraft. In RPGs, characters not controlled by players have become an integral part [4], which can help and guide to accomplish the goals of the game being independent of the player. For example, some characters are designed to play major or minor roles in game plots, and the player's interaction with them is typically part of advancing the narrative. Some others have no relationship to the plot, but their presence is meant to make the story world scenery richer and more believable. Such characters in games are important for enhancing the game experience, especially for the newly released games.

A majority of characters may look like the player; however, their behaviors clearly mark them as artificial and limited, remaining bland, robotic, and lifeless. Most of them offer limited interaction having to do with their specific purposes, which could break players' feeling of immersion. Besides, some characters that are not part of any plot or quest are barely distinguishable from the background scenery. It does not take long for game players to sense that the characters' behaviors are improbable. The lack of believability in the characters' design could result in a less meaningful game experience. Perceived believability is expected to increase players' feeling of immersion and their enjoyment [5]. Designing more human-like and more believable characters is likely to result in better role-playing experiences and further improving player retention. However, designing believable actions for hundreds of characters is cost-prohibitive for the industry.

As with the rapid development of machine learning techniques [6], we believe that significant automation and asset-reuse in this area are possible and hope to contribute to the design of actions and processes that allow more human-like and more believable characters. The ability to automatically designing human-like and believable characters for RPGs has a number of benefits. First, it can lessen the authorial burden of game designers, allowing renderings of a large number of characters, each with their own role and action patterns, all acting like human players. Second, game engagement can be enhanced as human-like, believable characters help maintain a sense of presence in the virtual world, and players' immersion is improved, especially for newly released games so that the player churn can further be avoided.

In order to design characters that act like human players, it is essential to learn behavior patterns from human players in RPGs. Behavior patterns are affected by many factors. First, the context (such as player levels and game classes) has an influence on the actions performed by players. For example, players in different levels tend to take different actions. Second, players take actions in order, and there exists a sequential relationship among actions, where the current action may be affected by the actions performed in the past

Manuscript received May 21, 2020; revised November 29, 2020; accepted January 13, 2021. This work was supported by the Natural Science Foundation of China under Grant 61802342, Grant 61802340, Grant 61772460, and Grant 61925603. *(Corresponding author: Gang Pan.)*

Sha Zhao, Yizhi Xu, Zhiling Luo, Shijian Li, and Gang Pan are with the Department of Computer Science, Zhejiang University, Hangzhou 310027, China (e-mail: szhao@zju.edu.cn; yzxu@zju.edu.cn; luozhiling@zju.edu.cn; shijianli@zju.edu.cn; gpan@zju.edu.cn).

Jianrong Tao and Changjie Fan are with the Fuxi AI Lab, NetEase Inc., Hangzhou 310052, China (e-mail: hztaojianrong@corp.netease.com; fanchangjie@corp.netease.com).

Digital Object Identifier 10.1109/TCSS.2021.3052261

and may influence the decision of what action to perform next. Third, some actions are closely related with each other, such as the pairwise actions and the actions frequently happening together. Recognizing such relationship benefits to infer the next most likely action.

In this article, we propose a novel framework to model player behaviors by integrating together the abovementioned factors, to learn behavior patterns of human players. On the basis of the learned behavior pattern, we generate human-like action sequences that can be used to design characters to enhance game engagement. Given only one action, our model generates an action sequence by predicting the succeeding action based on the previously generated actions. Our model is evaluated on a large scale data set of about 51 million action logs of QianNvYouHun, a computer RPG developed by NetEase Inc., collected from over 22 000 players during a time period of 21 days. The results show its effectiveness and stability. We investigate the generation performance for different game levels and find that the generation performance of the higher levels is better than that of the lower levels, suggesting that the context of the level has an influence on behavior patterns. It is also found that the difference in generation performance is slight when different actions are given, indicating the stability and robustness of our model. The contributions of our article are threefold.

1) We propose a novel framework for modeling player behaviors to learn behavior patterns, and on the basis of the behavior patterns learned from human players, it generates human-like action sequences given only one action. It is validated by a large-scale real-world data set and achieves state-of-the-art performance.
2) Our model includes actions and context (player levels and game classes) encoding, a long-term action memory (LTAM) working on the actions and context to learn hidden representations, and an action-attention mechanism to measure the contribution of the actions previously performed to the next action.
3) In addition to human-like action sequence generation, our model can be applied in many other applications, such as predicting the next action most likely to be performed by human players. Action prediction in games can support uploading game sources in advance, improving the game experience.

II. Related Work

In recent years, a growing number of studies have sought to model the behaviors of players in games. In the past, researchers have built behavior models of players in games using survey data or personal interviews, such as [7] and [8]; however, these techniques are time-consuming and result in models with limited generalizability [9]. As more research was done in this area, researchers began to incorporate data-driven techniques into their survey methodologies [10]–[16].

For example, Synnaeve and Bessiere [10] applied a Bayesian method to model a particular task that consists of choosing what to do and to select which target in a situation where allies and foes are present. The method learned the conditional probabilities from data gathered during human-player sessions in a massively multiplayer online role-playing game (MMORPG). Sharma *et al.* [11], [17] created a hybrid methodology by combining game traces along with player survey data in order to model player behaviors. They monitored players' progress through the game and matched the progress with a game trace from an accumulated library of traces. Mahlmann *et al.* [18] investigated whether one can predict specific aspects of player behaviors, examining the commercial game Tomb Raider: Underworld by supervised learning method. They forecast the moment at which a player will cease playing or how long the player would take to finish a game. Suznjevic *et al.* [19] performed action specific measurements of player sessions in terms of defined action categories for MMORPGs. They explored the hourly trends in user behavior and formed models based on observed patterns.

Gómez *et al.* [12] used variable-order Markov (VOM) to build a probabilistic model that is able to use the historic behavior of game players and to infer what will be their next actions. Liu *et al.* [14] worked out a method to predict player behaviors and movements in an educational game. The algorithm selects from a combination of methods-Markov models, state aggregation, and player heuristic search, according to whichever offers the largest amount of data. Lee *et al.* [15], likewise, developed a framework for predicting player movements within a pair of puzzle games by proposing a mixture of a one-depth heuristic search model and a data-driven Markov model. Valls-Vargas *et al.* [20] predicted a dynamic state of play, capturing fluctuations in player behaviors through the use of episodic information and time interval models within a sequential machine learning method capable of learning several models progressively.

There have been some studies that focus on analyzing sequential observed actions and modeling behaviors. For example, Wallner [16] proposed the use of lag sequential analysis that makes use of statistical methods to determine the significance of sequential transitions, so as to aid analysis of behavioral streams of players. Meaningful sequences involved in both frequent and rare patterns were extracted from the input data. Makarovych *et al.* [21] analyzed the sequential behavioral data and built profiles of players. They mined frequent sequences and then clustered the sequences to build behavioral profiles from the sequence data. Harrison and Roberts [9] used sequential observations to model and predict player behaviors. They first mined the actions of common co-occurrences and then monitored a player's action progress and determined the probability that the related actions will complete together.

There have been some studies modeling player behaviors using reinforcement learning [22], [23]. The studies, however, primarily focused on winning the game, such as achieving high scores or beating the opponents. The behavior patterns learned mostly ignored the rich and complex human motivations. Sometimes, human player actions are not directly related to the game's main objective, and players just want to have fun or enjoyment [24]. In addition, there have been some studies for goal recognition and plan recognition [25], [26]. Goal recognition seeks to discern one player's intentions by studying his or her actions [27] and plan recognition tackles

the more difficult challenge of predicting both a player's goal and the precise sequence of actions by which he or she will pursue it [28]. Goal/plan recognition is a closely related problem in which sequences of actions are used as input to predict the most likely goal/plan those actions that are trying to achieve. The main difference between our problem and plan/goal recognition is that we are predicting a sequence of actions directly rather than predicting goals and using those goals to predict actions.

III. Problem Description

Players usually play in populated environments with simple characters that are not controlled by players [29]. These characters have fixed and limited behaviors. They cannot learn from what experience in the game like a human player. It is easy for players to find them robotic and improbable, leading to player engagement decrease and even player churn. In order to promote game engagement, especially in the scenarios where characters are required, we model player's historical actions to learn behavior patterns and then, on the basis of the learned behavior patterns, generate a sequence of actions when an action is given. The behavior patterns of human players are preserved in the generated action sequences, where each action is selected based on the previously generated actions. Thus, the generated action sequences can be applied in designing characters, to make them more human-like and believable, promoting game engagement and supporting replayability.

Consider a role-playing game with the action set $\mathcal{A} = \{a_i\}_i^{|\mathcal{A}|}$ defined, a player u performs a set of his/her playing sequences $\mathcal{S}_u = \{s_i\}_i^{|\mathcal{S}|}$, where each s_i is the sequence of actions $(a_1, a_2, \ldots, a_k)$. Our goal is to generate human-like behaviors, i.e., action sequences, which can be used for designing game characters that are not controlled by players to enhance game engagement. This problem is formalized as follows.

Definition 1 (Behavior Generation): Given an action $a^* \in \mathcal{A}$ of a player u, generate his/her following behavior, i.e., an action sequence $\hat{s} = (\hat{a}_1, \hat{a}_2, \ldots)$, where $\hat{a}_i \in \mathcal{A}$.

This generated sequence $\hat{s}$ is considered as human-like if each action $\hat{a}_i$ follows the same behavior pattern of player u. It can be evaluated by computing the correspondence between the generated action sequences and the ground-truth action sequences of human players. For example, consider the player u having a sequence (a_1, a_2, a_3), and we can generate $(\hat{a}_2, \hat{a}_3)$ by taking a_1 as a^*. The closer $(\hat{a}_2, \hat{a}_3)$ is to (a_2, a_3), the better it is.

In order to generate human-like action sequences, it is essential to learn the behavior patterns of game players. There are some factors that have an influence on human player behaviors. First, the context of the current action performed by one player influences the decision of the follow-up actions. As players with different context progress a game, they are likely to make different types of actions. For example, players with different levels or game classes perform different actions. The varying of the context affects different behavior patterns of players in games. Second, the actions are performed in order by human players and form a sequence. By this nature, player's current actions may be affected by the actions performed in the past and may influence their upcoming actions. Thus, learning sequential relationship is necessary, where the decision of what action to perform next is influenced by the actions performed in the past. Third, some actions are close to each other, such as the pairwise actions, actions that co-occur frequently, and the ones with a causal relationship. Recognizing such a close relationship among actions benefits to discover which previous action is important to the next action so that it can be helpful to select the most likely action during the generation stage.

IV. Methods

In order to generate human-like action sequences, we propose a novel model that investigates together the factors mentioned above. First, we encode actions and their corresponding contexts, such as players' levels and game classes. Actions are encoded in dense representations, and the context is taken into consideration as well. The dense representations serve as input to an LTAM that is introduced to compute the hidden representation of actions and context. The LTAM cell features a sequence of memory blocks that include one memory cell, which outputs the hidden representation of actions and context. We then introduce an attention mechanism on the hidden representations. It adaptively weights the actions and learns the patterns from the input sequences. Based on the learned behavior patterns, given one action, our model predicts the succeeding action given actions previously generated and repeats one by one to generate a sequence of actions. Fig. 1 illustrates our model for action sequence generation.

A. Action and Context Encoding

The actions performed in the past by one player have an influence on the decision of what action to perform next. Besides the actions, the context when one player takes the action could influence the decision. Intuitively, one player takes actions depending on his/her context, such as the player's levels and game classes. Class is a job or profession commonly used to differentiate the abilities of different game characters, such as Assassin and Wizard. In an RPG, one player takes each action with a specific level and game class. Different levels usually lead to different actions due to different quests, even for the same player. Similarly, players with different game classes have different behavior patterns since available skills are different among classes. Thus, we take historical actions into consideration and the corresponding context as well. Here, we explore one player's level and game class, which are taken as the context.

Inspired by advances in distributed vector representations of text (e.g., words) [30], we encode each action in the sequence with a distributed representation. The corresponding level and game class of the same player are also encoded using distributed representations. To integrate the action, level, and game class together to learn behavior patterns, we concatenate the three embedded vectors. Formally, the input vector v_a is obtained by

$$v_a = [v_{\text{action}}, v_{\text{level}}, v_{\text{class}}] \tag{1}$$

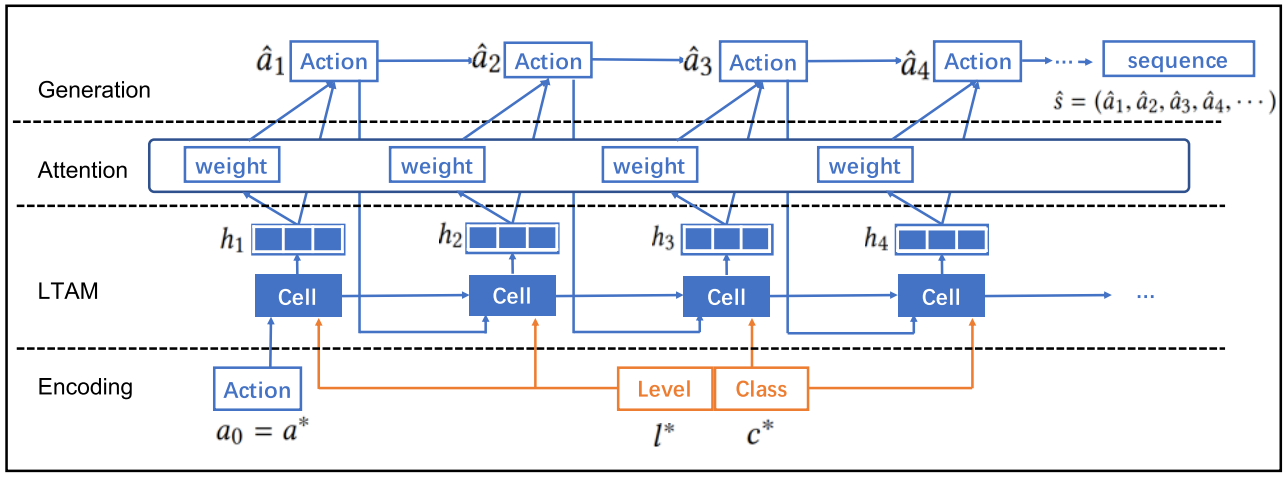


Fig. 1. Illustration of our model.

where [] represents the concatenation operation on vectors, and v_{action}, v_{level}, and v_{class} represent the distributed vectors of action a, level l, and game class c, respectively.

B. Long-Term Action Memory

Considering the sequential relationship among actions, it is important to learn behavior patterns from action sequences, in which the next action may be affected by the actions performed in the past. Extracting behavior patterns from sequence of player actions is likely to provide strong evidence to generate an action sequence. The model for learning sequential behavior patterns should robustly handle close relationships among the actions in a sequence, especially in a relatively long sequence. As a widely used recurrent neural network for solving time sequence problems, LSTM [31] leverages the cell state to keep the long-term memory in a sequence. Inspired by LSTM, we propose a component called LTAM. It preserves long-term lag capabilities to model players' behaviors and learns sequential behavior patterns. The LTAM features a sequence of memory blocks that include one memory cell to model the sequential relationship among historical actions in the sequence.

For the tth action a_t, the vector of a_t and the corresponding level l_t and game class c_t are concatenated as v_{a_t}, serving as input to LTAM. The memory cell includes three gating units: an input gate, a forget gate, and an output gate. In the implementation, the input gate q_t, forget gate f_t, and candidate memory cell $\tilde{c}_t$ for the tth action are computed in the following equation:

$$\begin{aligned} q_t &= \sigma(\boldsymbol{W}_t \cdot v_{a_t} + \boldsymbol{U}_t \cdot h_{t-1} + \boldsymbol{b}_i) \\ f_t &= \sigma(\boldsymbol{W}_f \cdot v_{a_t} + \boldsymbol{U}_f \cdot h_{t-1} + \boldsymbol{b}_f) \\ \tilde{c}_t &= \tanh(\boldsymbol{W}_c \cdot v_{a_t} + \boldsymbol{U}_c \cdot h_{t-1} + \boldsymbol{b}_c) \end{aligned} \tag{2}$$

where $\boldsymbol{W}$ and $\boldsymbol{U}$ are weight matrices for v_{a_t} and the cell output, i.e., hidden representation (h_{t-1}) for the $(t-1)$th input, $\cdot$ is dot product, $\boldsymbol{b}$ is the bias vector of each unit, and σ and tanh are the logistic sigmoid and hyperbolic tangent function.

A new state c_t is updated by modulating the current memory candidate value $\tilde{c}_t$ via the input gate q_t and the previous memory cell state (c_{t-1}) via the forget gate (f_t), as shown in (3). Through this process, a memory block decides whether to keep or forget the previous memory cell state via the forget gate and regulates the candidate of the current memory cell state via the input state

$$c_t = q_t \odot \tilde{c}_t + f_t \odot c_{t-1} \tag{3}$$

where $\odot$ is the cross product.

The output gate (o_t), similarly calculated as in (2), is utilized to compute the hidden representation (h_t) of the memory block for the tth input, based on the updated cell state (c_t) as follows:

$$\begin{aligned} o_t &= \sigma(\boldsymbol{W}_o \cdot v_{a_t} + U_o \cdot h_{t-1} + \boldsymbol{b}_o) \\ h_t &= o_t \odot \tanh(c_t). \end{aligned} \tag{4}$$

So far, we obtain h_t computed by the LTAM. h_t is the hidden representation of the tth action and the corresponding level and game class.

C. Attention on Hidden Representations

In a historical sequence, some actions are closely related to each other. For example, the entering and leaving the dungeon instance are pairwise related to each other, and the relationship between killing creatures and, thus, collecting the items from the killed creatures is a causal link. Recognizing such a relationship helps to discover which action is important for the next action so that we select the next most likely action when generating an action sequence. Here, we introduce an attention mechanism on hidden representations of actions to measure the contribution of each action to the next action and discover the relatively important action. The attention network [32] learns to assign attention scores to each word in the sentence, which, in other words, follows the diversity structure of data. In our work, the attention mechanism computes the weight of each action in the sequence for the next action, which makes each action have a different contribution. One action with a higher weight should be paid more attention to the next action. By introducing the attention mechanism, we can recognize which action is more important for the next action and further

infer players' intention from action sequences. Thus, attention is helpful for interpreting the generated sequence.

Formally speaking, the hidden representations $(h_1, h_2, \ldots, h_t)$ are integrated by a fully connected layer. It involves an affine parameter $\boldsymbol{W}_a$ and a bias parameter $\boldsymbol{b}_a$ and gets the attention weight $\boldsymbol{\chi}$. With the help of $\boldsymbol{\chi}$, the weighted sum of all the hidden state h_i at each step is used to compute the attention-based hidden state h_t^{ATT} in the softmax layer. Each weight χ_i is computed by

$$\chi_i = \tanh(\boldsymbol{W}_a[h_i, h_t] + \boldsymbol{b}_a). \tag{5}$$

With the attention mechanism, the attention-based hidden representation h_t^{ATT} is computed by (6), which is the weighted summation of h. Here, the superscript ATT is used to distinguish from the naive hidden representation h

$$h_t^{\text{ATT}} = \sum_{i=0}^{t} \chi_i h_i. \tag{6}$$

D. Generating Action Sequences

Once the attention-based hidden representation (h_t^{ATT}) is calculated for the tth action, it is used to predict the next most likely action a_t based on the previous t actions in a softmax layer, which is interpreted as a calculation of posterior probabilities of actions. The softmax function outputs the probability distribution over all candidate actions. The action that has the highest probability is considered as the one most likely to be performed next by one player. Given an action a^*, generate a sequence $\hat{s}$. With softmax, at generation phase, we have

$$\begin{aligned} \Pr(\hat{a}_t|\hat{a}_{t-1}, \ldots, \hat{a}_1, a_0 = a^*) &= \frac{e^{y_{\hat{a}_t}}}{\sum_i e^{y_i}} \\ \Pr(\hat{s}|a_0 = a^*) &= \prod_{t=1} \Pr(\hat{a}_t|\hat{a}_{t-1}, \ldots, \hat{a}_1, a_0 = a^*). \end{aligned} \tag{7}$$

Each of y_i is unnormalized probability for each predicted $\hat{a}_t$, computed as

$$y = Z^T h_t^{\text{ATT}} \tag{8}$$

where Z is the parameter of softmax and h_t^{ATT} is the attention-based hidden representation of the tth action.

The objective of the model in the training procedure is to maximize the cross entropy

$$\frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \Pr(\hat{s}|a_i) \log \Pr(\hat{s}|a_i) \tag{9}$$

where $\mathcal{S}$ is the set of all sequences.

E. Model Training

In order to train our model, we need to estimate the parameters, including $\boldsymbol{W}_t$, $\boldsymbol{U}_t$, $\boldsymbol{W}_f$, $\boldsymbol{U}_f$, $\boldsymbol{W}_c$, $\boldsymbol{U}_c$, $\boldsymbol{W}_o$, and $\boldsymbol{U}_o$ in the LTAM, $\boldsymbol{W}_a$ and $\boldsymbol{b}_a$ in attention on hidden states, and Z^T in the softmax layer. The parameters can be summarized as $\Theta = (\boldsymbol{W}_t, \boldsymbol{U}_t, \boldsymbol{W}_f, \boldsymbol{U}_f, \boldsymbol{W}_c, \boldsymbol{U}_c, \boldsymbol{W}_o, \boldsymbol{U}_o, \boldsymbol{W}_a, \boldsymbol{b}_a, Z)$. The objective function of the model in the training phase is the negative cross entropy on conditional probability of the next action with respect to an observed action sequence, as shown in the following equation:

$$\arg\min_{\Theta} -\frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \Pr(\hat{s}|a_i) \log \Pr(\hat{s}|a_i). \tag{10}$$

Each instance of training data consists of a sequence of actions. For example, if three actions, a_1, a_2, and a_3, are taken as input to the model, there data examples are generated: 1) $[a_1]$ for a_2; 2) $[a_1, a_2]$ for a_3; and 3) $[a_1, a_2, a_3]$ for the next action a_4.

The action vectors are initialized randomly at the beginning and embedded as a vector of G dimensionality. Each action vector is trained using a stochastic gradient, and the gradient is obtained via backpropagation through time (BPTT). At every step of stochastic gradient descent, one can sample a fixed-length sequence of a random player, compute the error gradient from the network recurrently until all the actions in the sequence are input and use the gradient to update action vectors and softmax parameter Z. During the training procedure, cross entropy is utilized for the loss function.

The model is trained to generate action sequences. In the generation procedure, an action a^* is given at the beginning, and the next action is predicted based on the given action. Then, the model predicts the succeeding action based on the generated prefix. For example, if given an action a_t, the action a_{t+1} is predicted based on the action a_t by the model, the action a_{t+2} is predicted based on the generated action a_{t+1}, and so on.

(a) (b)

Fig. 2. Two game scene examples of Qiannvyouhun. (a) Two game roles are combating with each other at an arena. (b) Game player is shopping in a virtual market.

V. Data Set Overview

We evaluated our model with a large-scale real-world data set collected by NetEase Inc., a Chinese Internet technology company. As one of the largest Internet and video game company, NetEase Inc. develops and operates online PC and mobile games. The data set that we used contains action logs of QianNvYouHun,[1] a computer game developed by NetEase Inc.[2] Qiannvyouhun is a multiple RPG that is designed on the basis of a love story between a folk people and a female ghost. There are many game scenes designed in the game, such as combating with other players and shopping in a virtual market, as shown in Fig. 2.

[1]http://xqn.163.com
[2]http://game.163.com

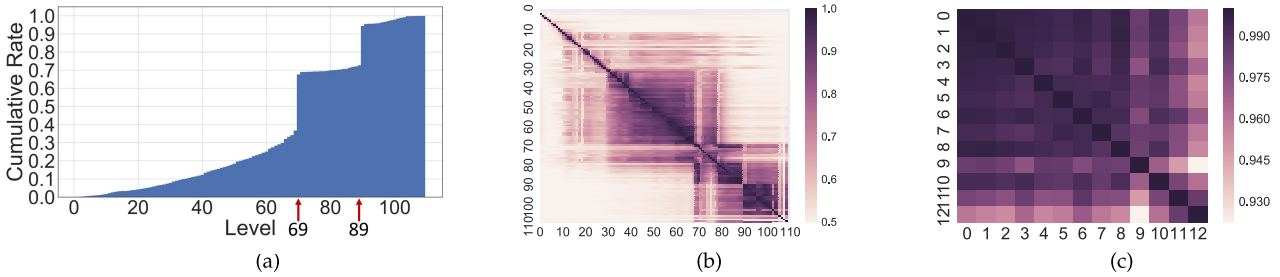


Fig. 3. Basic analysis of the data set. (a) CDF of action records in terms of levels. (b) Confusion matrix of different levels. (c) Confusion matrix of different classes.

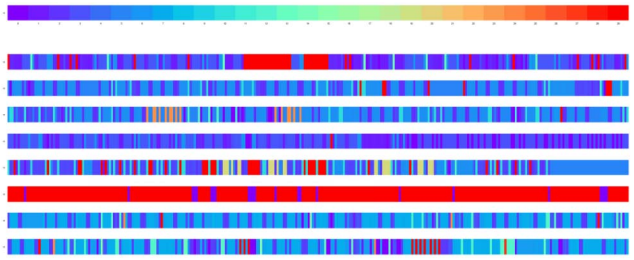


Fig. 4. Few action sequences in Qiannvyouhun.

In the data set, there are more than 424 million records from more than 81 000 players during the time period from September 21, 2018, to October 11, 2018. Each record consists of: 1) identification (ID) of one player (anonymized); 2) the time stamp when an action is performed; 3) action ID; 4) the description of the corresponding action; 5) the level of the player when he/she performs the action; and 6) the game class of the player when performing the action. The data set does not contain any personally identifiable information, where the "player ID" has been anonymized and does not contain any player metadata. All the researchers are regulated by the strict nondisclosure agreement, and the data set is located in a secure off-line server.

In the data set, there are many types of actions, such as log in, log out, completing quests, interactions, entering a dungeon, trading, and chatting or communicating with other players. In order to give a better understanding of the data set, we present some action sequences in Fig. 4, where each slot represents action and each type of action is indicated by one color. In Fig. 4, each action sequence consists of 300 continuous actions, where the actions are performed one by one. It can be seen that most behavior sequences are composed of different types of actions, except the third one from the end, where most actions are in red. We found that the red color here indicates the social action type, such as chatting or communicating with other players. In other words, the third sequence from the end indicates that one player chats or communicates frequently with others but rarely performs other types of actions. For the other action sequences, the actions in purple and blue color occur frequently in the sequences, which represents the actions of completing quests and killing creatures, respectively.

To use the data set for evaluation, we performed a preprocessing work.

Extracting Key Actions: When we refer to player actions, we were primarily concerned with the key actions players perform. For example, completing a particular set of quests is a sequence of "actions" and of interest for building our behavior model. Lower level actions that are dispensable, such as using some kinds of skills, jumping, or running, however, are not taken into consideration. In other words, the actions we were concerned with is when a significant game event is caused by the player (either by atomic actions or through a sequence of actions) and can be interpreted as an attribute or indicator of the game content the player experiences. For example, the actions of entering the dungeon, killing creatures, collecting items from the corpses of creatures that have been killed, and, finally, leaving the dungeon represent the game event of completing a dungeon quest. After extracting key actions, there were 8904 key actions in total.

Filtering Players: Our study focused on the players who have relatively rich action records, and we removed the players whose key action records were less than 1000. Overall, there are about 22 000 players remained and about 51 million key action records in total, which were used in the following experiments.

Basic Analysis: We performed basic analysis on the data set after preprocessing. During the data collection period, players level up progressively. According to our observation, players reach different final levels till the end of the data collection. There are 109 levels in total in the game and 13 game classes. We computed the cumulative distribution function of the action logs in terms of game levels, as shown in Fig. 3(a). It can be seen that Level 69 and Level 89 are two distinct boundaries, for which there are relatively rich action logs.

We also investigated the cosine similarity among actions in different levels [see Fig. 3(b)] and among different game classes as well [see Fig. 3(c)], to observe the differences in how actions distribute in different levels or classes. Here, each level was represented as a vector of 8904 dimensionality, where each dimension was one action and its value was the occurrences of the action in the level. Even for the same action, the occurrences are significantly different at different levels. Fortunately, cosine similarity is not sensitive to the absolute values of vectors. The cosine similarity was computed between any two levels, as shown in Fig. 3(b). As we can see, the levels from 30 to 80 and the levels from 70 to 100 have similar action distributions. However, for the low levels, such as from 0 to 10, the action distributions are a little different. It may be because

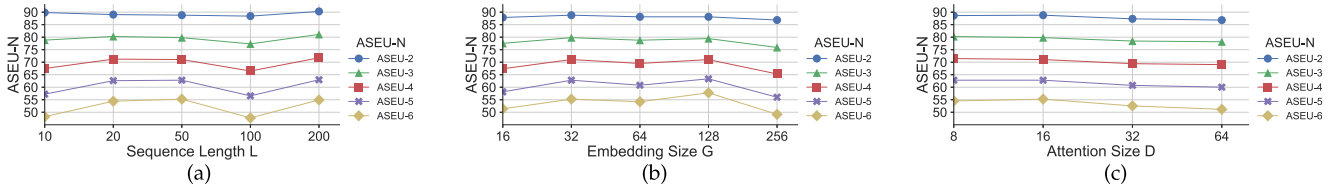


Fig. 5. Generation performance with varying (a) sequence length, (b) embedding size, and (c) attention size.

player characters gain new skills as they level up, and they use different skills at different levels. Besides, for the levels that are a little far away from each other, such as a low Level 1 and a high Level 100, the action distributions are different.

We also computed the cosine similarity between any two classes that were similarly represented using the occurrences of each action, as shown in Fig. 3(c). For most game classes, the action distributions are similar. Among all the classes, Class 9 and Class 12 have the greatest difference in action distributions. Class 9 is Cleric, a healer, who has powers to heal wounds, protects their allies, and, sometimes, resurrects the dead, while Class 12 is Assassin who usually has attacks that cause a high amount of damage in a short amount of time.

VI. Experiments

In this section, we conducted extensive experiments to show the effectiveness of our model. We first describe the performance metric used in the experiments.

A. Experiment Setup

1) Performance Measurement: We defined a metric to measure the generation performance, called action sequence evaluation understudy (**ASEU**), inspired by the idea of bilingual evaluation understudy (BLEU) [33]. BLEU is an algorithm for evaluating the quality of the text that has been machine-translated from one natural language to another. For a generated action sequence, quality is considered to be the correspondence between an action sequence generated by our model and that of a human player: "the closer a sequence generation is to a human player, the better it is"—this is the central idea behind ASEU.

We first computed the geometric average of n-gram precision [33], p_n, by (11). n-gram is a contiguous sequence of n actions from a given sequence. For example, for a generated sequence consisting of 20 actions, the number of 2-gram, 3-gram, and 6-gram is 19, 18, and 15, respectively. We counted the number of the n-gram that can be matched in the references and divided it by the total number of the candidate n-grams to get the n-gram precision. If there are 10 that can be matched in the references out of the total 19 2-grams, the 2-gram precision is 10/19

$$p_n = \frac{\sum_{i=1}^{m-(n-1)} \mathbf{1}^{(\text{n-gram=true})}}{m-(n-1)} \tag{11}$$

where $\mathbf{1}(\cdot)$ is an indicator function, m is the length of the generated action sequence, $(m-(n-1))$ is the number of the total n-grams in the generated action sequence of m length, and (n − gram = true) means that the candidate n-gram can be matched in the references. Then, ASEU-N can be computed by

$$\text{ASEU–}N = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \times 100 \tag{12}$$

where w_n is the positive weights, and N is the maximum value of n-gram.

In our work, we tested $N = 2, 3, 4, 5,$ and 6, respectively, and used uniform weights $w_n = 1/N$.

Given the same action, the model can generate many action sequences. Thus, we used ***self-ASEU*** to evaluate the diversity of the generated behavioral sequences, inspired by self-BLEU [34]. ASEU can be used to evaluate how one action sequence resembles the rest in a generated collection. Regarding one generated sequence as the hypothesis and the others as references, we calculated the ASEU score for every generated sequence and defined the average ASEU score to be the self-ASEU of the generated sequences. A lower self-ASEU score means that the generated sequences are more diverse given the same action.

2) Compared Approaches: We selected the following algorithms for comparison.

1) Random refers to predicting the succeeding action by randomly selecting one action from all the candidate actions based on their occurrences in the data set, to generate a sequence.
2) HMM [35] forms a Markov chain with hidden states that are mapped to the observed action, and the probability distribution of the observed action depends on the hidden states.
3) SeqGAN [36] is a discrete RNN-based generator. We also evaluated LeakGAN [37], another discrete RNN-based generator, but, since it is similar to seqGAN and has lower performance and higher computation cost, we omit it for brevity.
4) Transformer [38] is a network architecture based solely on attention mechanisms, dispensing with recurrence and convolutions entirely.
5) LSTM-Transformer applies LSTM on the input vectors and then takes the cell output as the input to the transformer.

B. Results and Analysis

1) Performance Study With Respect to Sequence Length, Embedding Size, and Attention Size: We first investigated the performance of our model with varying sequence length L, embedding size G, and attention size D, as shown in Fig. 5. The sequence length L refers to the number of actions

TABLE I
ASEU SCORE COMPARISON WITH OTHER APPROACHES

Method	ASEU-2	ASEU-3	ASEU-4	ASEU-5	ASEU-6
Random	14.05	4.92	2.81	2.05	1.68
HMM	52.76	24.50	11.33	8.90	7.87
Transformer	67.42	52.64	41.16	31.22	24.03
LSTM-Transformer	85.65	71.35	57.93	46.95	38.53
SeqGAN	92.88	83.28	72.12	61.26	51.58
Our model	**96.49**	**91.60**	**85.68**	**79.30**	**72.65**

TABLE II
SELF-ASEU SCORE COMPARISON WITH OTHER APPROACHES

Method	selfASEU-2	selfASEU-3	selfASEU-4	selfASEU-5	selfASEU-6
HMM	59.92	28.26	14.70	10.44	8.76
Transformer	39.50	10.86	5.25	3.44	0.26
LSTM-Transformer	81.33	65.41	50.87	38.40	31.07
SeqGAN	92.28	83.59	73.82	64.17	55.23
Our Model	89.45	80.65	71.26	62.26	54.26

in a sequence input to the model during the training procedure, embedding size G refers to the dimensionality of each combination vector of action vector, level vector, and class vector, and attention size D refers to the number of neurons in the attention network. As we can see from Fig. 5(a), the ASEU scores increase when using ten actions in a sequence to 20 actions, slightly change from 20 actions to 200 actions, and even slightly decrease when using 100 actions. In Fig. 5(b), there is an improvement in ASEU scores when using 16 dimensions to 32 dimensions and a slight change from 32 dimensions to 128 dimensions but a decline when using 256 dimensions to represent the combination vector. We also tested the generation performance with varying attention size, as shown in Fig. 5(c). The ASEU slightly changes when varying the attention size from 8 to 64.

In the following experiments, we set the embedding size to 32, the attention size to 16, and the sequence length to 20. In other words, the combination vector of action, level, and the class was represented as a vector of 32 dimensionality. In the attention network, we used 16 neurons. We sliced the action records into sequences, and each sequence consists of 20 actions. In the training procedure, we used 50 000 sequences each of which consisting of 20 actions to train the model, among which 40 000 sequences were used as the training set and the rest 10 000 used as the validation set. We used another 50 000 sequences as the test set, each of which consists of 20 actions. In the training procedure, we employed the fivefold cross-validation policy.

2) Comparison With Other Approaches: We compared our model with other approaches, as shown in Table I. We built an HMM with 64 hidden states, a seqGAN, a transformer (16 heads and three hidden layers), and an LSTM-Transformer to generate action sequences, respectively. All of the approaches were trained using sequences consisting of 20 actions. In practice, we tuned different parameters for each approach to achieve the best performance. For all the approaches, in the generation procedure, we generated sequences consisting of 20 actions. The length of the generated sequences is the same as that of the reference sequences in the test data. For a given action, we generated 2000 sequences, and the ASEU score is the average value. Here, the references are all the sequences in the test data when we computed ASEU scores.

As shown in Table I, for all the approaches, ASEU-N decreases as N increases, and ASEU-2 is the best performance, while ASEU-6 is the worst. It is obvious that it is more difficult to achieve high performance when generating a longer action sequence. As we can see from Table I, our model performs the best in any ASEU-N ($N = 2, 3, 4, 5, 6$), and our model > seqGAN > LSTM-transformer > Transformer > HMM > Random. As N increases, the improvement of our model is more significant compared with that of other approaches. In particular, our model achieves an ASEU-6 score of 72.65, 71 higher than Random (72.65 versus 1.68), about nine times higher than HMM (72.65 versus 7.87), almost three times the ASEU-6 of Transformer model (72.65 versus 24.03), 34 higher than that of LSTM-Transformer (72.65 versus 38.53), and 21 higher than that of seqGAN (72.65 versus 51.58). This suggests the distinct advantage of our model in generating a longer behavior sequence.

In generating shorter sequences, our model still outperforms the other approaches. For example, our model achieves an ASEU-2 score of 96.49, 44 higher than that of HMM (96.49 versus 52.76), 30 higher than Transformer (96.49 versus 67.42), 10 higher than LSTM-Transformer (96.49 versus 85.65), and 4 higher than seqGAN ((96.49 versus 92.88). In particular, our model produces a much better performance than LSTM-Transformer, which is an LSTM-based model. As we mentioned in the related work part, some existing studies on goal/plan recognition applied LSTM-based models. The comparison indicates that our model has advantages in generating action sequences.

We also reported the self-ASEU (see Table II) to measure the diversity of the 2000 sequences for a given action. A lower self-ASEU score means a higher diversity. For the diversity of generated sequences, the transformer performs better than ours, but its ASEU scores are much worse than ours.

In order to investigate the stability and robustness of our model, we computed the performance difference of our model when given different actions to generate action sequences. We randomly selected 100 actions from all the 8904 key actions as given actions for generation and generated action sequences, respectively. We computed the variance when given different actions: 1.08E-05, 1.45E-05, 7.84E-05, 1.09E-04, 1.25E-04 in ASEU-2, ASEU-3, ASEU-4, ASEU-5, and ASEU-6. The variance is very small, suggesting that there is a very light difference in generation performance when different actions are given and out model is stable and robust.

3) Effectiveness of Different Components: We investigated the effectiveness of the components to performance, including attention mechanism and context (level and class), as shown in Table III. To be specific, we took LTAM (long short action memory) as a baseline model, in which only actions were input without considering the context. We added context

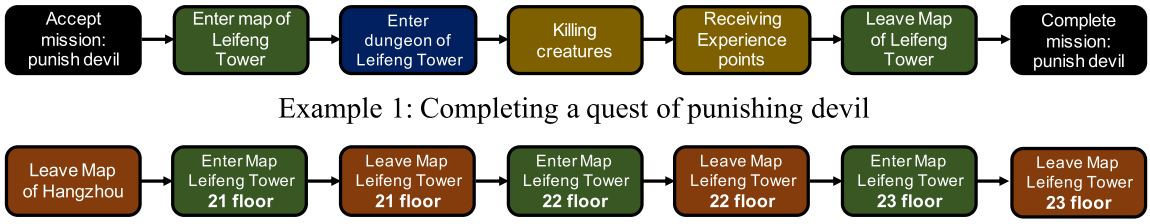


Fig. 6. Two examples of the generated action sequences.

TABLE III
EFFECTIVENESS OF DIFFERENT COMPONENTS

LTAM (Baseline)	ASEU-2	82.95	ASEU-4	60.70	ASEU-6	40.43
LTAM+context	ASEU-2	91.63	ASEU-4	78.59	ASEU-6	65.30
	Improvement	**8.68**	Improvement	**17.89**	Improvement	**23.87**
LTAM+attention	ASEU-2	83.85	ASEU-4	62.42	ASEU-6	42.45
	Improvement	**0.90**	Improvement	**1.72**	Improvement	**2.02**
Ours	ASEU-2	92.29	ASEU-4	79.96	ASEU-6	65.65
	Improvement	**9.34**	Improvement	**19.26**	Improvement	**25.22**

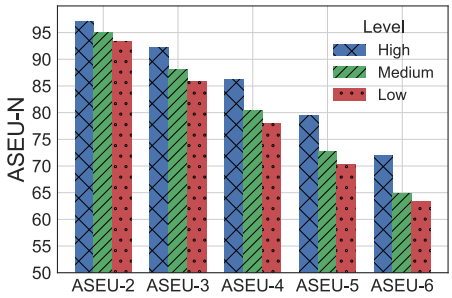


Fig. 7. Performance with respect to different game levels.

and attention mechanism to the baseline model and then compared the ASEU-N ($N = 2, 4, 6$) with the baseline model, respectively. "Ours" refers to the model that combines LTAM, context, and attention mechanism together. Here, the context involved here refers to Level 69 and Class 0, and the sequences of Level 69 and Class 0 in the test data were used as references when computing the ASEU score. As we can see from Table III, our model performs much better than the baseline LTAM, further indicating that our model has advantages in generating actions. Adding context of the actions makes the greatest contribution to generating action sequences. Compared to the baseline model that only takes actions as input, the model that takes player levels and game classes into consideration makes the performance 8.68, 17.89, and 23.87 higher in ASEU-2, ASEU-4, and ASEU-6, respectively. The longer the sequence generation, the more significant the improvement. The comparison further validates the importance of the context in modeling game player behaviors.

The attention mechanism contributes 0.90, 1.72, and 2.02 in ASEU-2, ASEU-4, and ASEU-6, respectively, compared with the baseline model. Although the attention mechanism cannot improve the performance as significant as adding context, it can discover which action is important for the next action, which can help interpret the generation results. Compared to the baseline model, combining the game context and attention mechanism significantly improves the performance, 9.34, 19.26, and 25.22 higher than in ASEU-2, ASEU-4, and ASEU-6, respectively. Our model comprehensively considers the sequential relationship among actions, game context, and the contribution of different actions in modeling player behavior patterns, which achieves state-of-the-art performance in action sequence generation.

4) Performance With Respect to Different Levels: As we can see from Fig. 3(a), Level 69 and Level 89 are two distinct boundaries. The actions are different at different levels, as shown in Fig. 3(b). Thus, we investigated the generation performance with respect to different levels. We divided the game level into three categories, low [1, 69), medium [69, 89), and high [89, 109], and trained three-generation models using the action logs in the corresponding level category. Fig. 7 shows the performance comparison in different levels. It can be seen that all the ASEU scores in high levels are higher than those of medium and low levels, and all the ASEU scores in medium levels are higher than those of low levels. The higher the level is, the better the performance. It suggests that actions performed by players with higher levels tend to follow some patterns. One player with a higher level is more familiar with the game, and he/she tends to take specific actions following his/her pattern when completing a quest.

5) Illustration of Generated Sequences: In order to better understand the generated action sequences, we inspected two examples from the results, as shown in Fig. 6, each of which consists of seven actions. The first example is about completing the quest of punishing the devil. The action sequence starts with accepting the mission. To complete the quest, it needs to enter the map of Leifeng Tower and enters a dungeon. In the dungeon, creatures are killed, and experience points are received. Then, it leaves the map of Leifeng Tower, and finally, the mission is completed. In this generated action sequence, the pairwise relationship between entering and leaving and the causality between first killing creatures and subsequently receiving experience points are well preserved. In the second example, the generated action sequence contains entering and leaving Leifeng Tower floor by floor, from the 21st to the 23rd floor. The floors are accessed in order from the lower ones to the higher ones. Both two examples show the capability of performing human-like actions of our generation model.

6) Performance on Different Amounts of Training Data: We also investigated the performance of our model trained

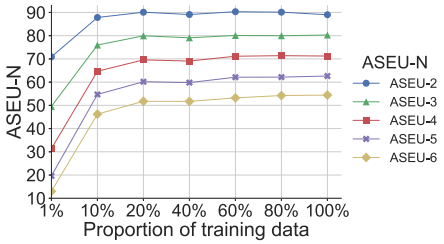


Fig. 8. Performance with partial data for training.

by different amounts of training data (1%, 10%, 20%, 40%, 60%, and 80%). Training data 10%, for example, mean that we randomly selected 10% of the action records from the whole training data set to train the model. We just varied the training data size but kept the testing data as before. The random selection of the training data was carried out five times independently. As we can see from Fig. 8, our model still performs well when only 10% of training data set, achieving an ASEU-6 score of 46. The performance increases quickly when 1%–10% of the training data were used. There is a slight difference in performance when 20%–100% of the training data were used. It shows that our model performs stably with varying amounts of training data even though the small size of the training data was used for training.

VII. Conclusion

In this work, we have proposed a novel framework to model the behaviors of RPG players. Given only one action, the model can generate human-like action sequences, which can be used to design characters that are not controlled by players to enhance game engagement and further avoid player churn. The framework includes actions and context encoding, an LTAM to learn the hidden representations, and an attention mechanism on hidden representations. The model generates action sequences by predicting the succeeding action given the actions previously generated. Extensive experiments were conducted on a real-world data set of over 51 million action records from about 22 000 game players in 21 days. The generation results demonstrate the state-of-the-art performance, outperforming the other approaches.

Although we have effectively modeled player behaviors, we must acknowledge that there are still some limitations. First, our data set used in the experiment is limited to only one game. Whether our model could be generalized to other RPGs is still unknown. Second, our model performs well in ASEU-N, especially when N is relatively small. Its performance needs to be improved when N is bigger. Third, our model has not been deployed in practice yet. In future work, we will address the abovementioned problems, such as collecting records from other RPGs to evaluate our model, improving the performance in generating relatively long action sequences and implementing our model online for practical applications. Moreover, we can apply our model for action prediction. Our model can not only generate action sequences but also predict the next action of players. Predicting the next action most likely to be performed by players is important for game companies. If players' actions can be predicted, the game company can prepare game sources in advance, such as uploading maps and highlighting the game equipment players need next. It can reduce game delay and improve player experience in games.

References

[1] M. Suznjevic and M. Matijasevic, "Player Behavior and traffic characterization for MMORPGs: A survey," *Multimedia Syst.*, vol. 19, no. 3, pp. 199–220, Jun. 2013.

[2] J. Tao, J. Xu, L. Gong, Y. Li, C. Fan, and Z. Zhao, "NGUARD: A game bot detection framework for NetEase MMORPGs," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 811–820.

[3] A. L. Jia, S. Shen, R. V. D. Bovenkamp, A. Iosup, F. Kuipers, and D. H. Epema, "Socializing by gaming: Revealing social relationships in multiplayer online games," *ACM Trans. Knowl. Discovery Data (TKDD)*, vol. 10, no. 2, p. 11, 2015.

[4] G. Pickett, F. Khosmood, and A. Fowler, "Automated generation of conversational non player characters," in *Proc. AIIDE*, 2015, pp. 92–97.

[5] K. Bhatt, "Believability in computer games," in *Expertise in Design, Design Thinking Research Symposium*. Sydney, NSW, Australia: University of Technology, Nov. 2003, p. 81.

[6] F.-Y. Wang, "Computational dissemination: Toward precision and smart impacts for computational social systems," *IEEE Trans. Comput. Social Syst.*, vol. 4, no. 4, pp. 193–195, Dec. 2017.

[7] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," *J. MUD Res.*, vol. 1, no. 1, p. 19, 1996.

[8] N. Yee, "Motivations for play in online games," *Cyberpsychol. Behav.*, vol. 9, no. 6, pp. 772–775, Dec. 2006.

[9] B. Harrison and D. L. Roberts, "Using sequential observations to model and predict player behavior," in *Proc. 6th Int. Conf. Found. Digit. Games - FDG*, 2011, pp. 91–98.

[10] G. Synnaeve and P. Bessiere, "Bayesian modeling of a human MMORPG player," *AIP Conf. Proc.*, vol. 1305, no. 1, pp. 67–74, 2011.

[11] M. Sharma, S. Ontañón, M. Mehta, and A. Ram, "Drama management and player modeling for interactive fiction games," *Comput. Intell.*, vol. 26, no. 2, pp. 183–211, May 2010.

[12] A. Baldominos Gómez, E. A. García, I. Marrero, and Y. S. Achaerandio, "Real-time prediction of gamers behavior using variable order Markov and big data technology: A case of study," *Int. J. Interact. Multimedia Artif. Intell.*, vol. 3, no. 6, pp. 44–51, 2016.

[13] Y. Xiao, J. Li, Y. Zhu, and Q. Li, "User Behavior prediction of social hotspots based on multimessage interaction and neural network," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 2, pp. 536–545, Apr. 2020.

[14] Y.-E. Liu *et al.*, "Predicting player moves in an educational game: A hybrid approach," in *Proc. Educ. Data Mining*, 2013, pp. 106–113.

[15] S. J. Lee, Y.-E. Liu, and Z. Popovic, "Learning individual behavior in an educational game: A data-driven approach," in *Proc. Educ. Data Mining*, 2014, pp. 114–121.

[16] G. Wallner, "Sequential analysis of player behavior," in *Proc. Annu. Symp. Comput.-Human Interact. Play*, Oct. 2015, pp. 349–358.

[17] M. Sharma, S. Ontanón, C. R. Strong, M. Mehta, and A. Ram, "Towards player preference modeling for drama management in interactive stories," in *Proc. FLAIRS Conf.*, 2007, pp. 571–576.

[18] T. Mahlmann, A. Drachen, J. Togelius, A. Canossa, and G. N. Yannakakis, "Predicting player behavior in Tomb Raider: Underworld," in *Proc. IEEE Conf. Comput. Intell. Games*, Aug. 2010, pp. 178–185.

[19] M. Suznjevic, I. Stupar, and M. Matijasevic, "MMORPG player behavior model based on player action categories," in *Proc. 10th Annu. Workshop Netw. Syst. Support Games*, Oct. 2011, p. 6.

[20] J. Valls-Vargas, S. Ontanón, and J. Zhu, "Exploring player trace segmentation for dynamic play style prediction," in *Proc. AIIDE*, 2015, pp. 93–99.

[21] S. Makarovych, A. Canossa, J. Togelius, and A. Drachen, "Like a dna string: Sequence-based player profiling in Tom Clancy's the division," in *Proc. AIIDE*, 2018, pp. 1–9.

[22] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[23] G. Kumar, M. Henderson, S. Chan, H. Nguyen, and L. Ngoo, "Question-answer selection in user to user marketplace conversations," 2018, *arXiv:1802.01766*. [Online]. Available: http://arxiv.org/abs/1802.01766

[24] B. Wang, T. Sun, and X. S. Zheng, "Beyond winning and losing: Modeling human motivations and behaviors using inverse reinforcement learning," 2018, *arXiv:1807.00366*. [Online]. Available: http://arxiv.org/abs/1807.00366
[25] W. Min, B. W. Mott, J. P. Rowe, B. Liu, and J. C. Lester, "Player goal recognition in open-world digital games with long short-term memory networks," in *Proc. IJCAI*, Jul. 2016, pp. 2590–2596.
[26] F. Bisson, H. Larochelle, and F. Kabanza, "Using a recursive neural network to learn an agent's decision model for plan recognition," in *Proc. IJCAI*, 2015, pp. 918–924.
[27] S. Carberry, "Techniques for plan recognition," *User Model. User-Adapted Interact.*, vol. 11, nos. 1–2, pp. 31–48, 2001.
[28] D. Hooshyar, M. Yousefi, and H. Lim, "Data-driven approaches to game player modeling: A systematic literature review," *ACM Comput. Surv.*, vol. 50, no. 6, p. 90, 2018.
[29] T. Rhujittawiwat and V. Kotrajaras, "Learnable buddy: Learnable supportive ai in commercial MMORPG," in *Proc. Int. Conf. Comput. Games, AI, Animation, Mobile, Educ. Serious Games*, 2006, pp. 1–5.
[30] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.
[31] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[32] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473
[33] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proc. ACL*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 311–318.
[34] Y. Zhu *et al.*, "Texygen: A benchmarking platform for text generation models," 2018, *arXiv:1802.01886*. [Online]. Available: http://arxiv.org/abs/1802.01886
[35] A. B. Poritz, "Hidden Markov models: A guided tour," in *Proc. IEEE Conf. Acoust., Speech Signal Process. (ICASSP)*, 1988, pp. 7–13.
[36] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proc. AAAI*, 2017, pp. 2852–2858.
[37] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," in *Proc. AAAI*, 2018, pp. 5141–5148.
[38] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

Sha Zhao (Member, IEEE) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2017.

She visited Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA, USA, as a Visiting Ph.D. Student, from 2015 to 2016. She is currently a Post-Doctoral Research Fellow with the College of Computer Science and Technology, Zhejiang University. Her research interests include pervasive computing, data mining, and machine learning.

Dr. Zhao received the Best Paper Award of ACM UbiComp16.

Yizhi Xu received the B.Sc. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2017, where he is currently pursuing the master's degree with the College of Computer Science and Technology.

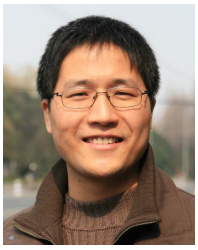
His research interests include machine learning and data mining.

Zhiling Luo received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2012 and 2017, respectively.

He was an Assistant Professor of computer science with Zhejiang University. He was the Visiting Scholar with the Georgia Institute of Technology, Atlanta, GA, USA, in 2016. His research interests include service computing, machine learning, and data mining.

Jianrong Tao (Member, IEEE) received the B.Sc. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2014, and the master's degree in computer science from Zhejiang University, Hangzhou, China, in 2017.

He is currently an Algorithm Expert with the Fuxi AI Lab, NetEase Inc., Hangzhou. His research interests include machine learning, data mining, network analysis, and user profiling.



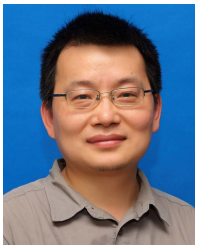
Shijian Li received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2006.

In 2010, he was a Visiting Scholar with the Institute Telecom SudParis, Évry, France. He is currently with the College of Computer Science and Technology, Zhejiang University. He has published over 40 articles. His research interests include sensor networks, ubiquitous computing, and social computing.

Dr. Li also serves as a reviewer or a PC member of more than ten conferences. He also serves as an Editor for the *International Journal of Distributed Sensor Networks*.

Changjie Fan received the B.S. and Ph.D. degrees in computer science from the University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively.

He is currently the Director of the FUXI AI Lab, NetEase Inc., Hangzhou, China. His research interests include multiagent systems, reinforcement learning, game theory, and knowledge discovery.



Gang Pan (Member, IEEE) received the B.Eng. and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1998 and 2004, respectively.

From 2007 to 2008, he was a Visiting Scholar with the University of California at Los Angeles, Los Angeles, CA, USA. He is currently a Professor with the Department of Computer Science and the Deputy Director of the State Key Lab of CAD&CG, Zhejiang University. His current interests include artificial intelligence, pervasive computing, brain-inspired computing, and brain–machine interfaces.