Journal of Advanced Computational Intelligence and Intelligent Informatics, Special issue on Advances in Intelligent Data Processing, vol. 11, no. 3, pp. 319-326, 2007.

Classification of Online Game Players Using Action Transition Probability and Kullback Leibler Entropy

Ruck Thawonmas* **and Ji-Young Ho*,****

*Intelligent Computer Entertainment Laboratory, Graduate School of Science and Engineering
Ritsumeikan University, Kusatsu, Shiga 525-8577, Japan
E-mail: ruck@ci.ritsumei.ac.jp
**SAMSUNG ELECTRONICS CO.,LTD.
[Received 00/00/00; accepted 00/00/00]

Online game players are more satisfied with contents tailored to their preferences. Player classification is necessary for determining which classes players belong to. In this paper, we propose a new player classification approach using action transition probability and Kullback Leibler entropy. In experiments with two online game simulators, Zereal and Simac, our approach performed better than an existing approach based on action frequency and comparably to another existing approach using the Hidden Markov Model (HMM). Our approach takes into account both the frequency and order of player action. While HMM performance depends on its structure and initial parameters, our approach requires no parameter settings.

Keywords: player classification, online game, data mining, customer relationship management, game design

1. Introduction

The game industry is expanding rapidly due to computer advances and growing social acceptance, especially online games [1]. Due to high competition in the online-game industry, it is crucial to provide suitable game content for individual players or groups to ensure high user satisfaction. Similar to other e-Commerce business, data mining plays an important role in understanding player behavior.

In a new data-mining application called game mining (**Fig. 1**) [2], players are classified based on appropriately selected input features from game logs, and content is provided to them based on their personal behavior. Another research group independently proposed a data-mining application to online-game task syntheses [3], where such tasks are dynamically assigned to individual players. In both, classification is a key technology.

We previously proposed two approaches to player classification, one using the normalized action frequency vector (NAFV) [2] and the other using the Hidden Markov Model (HMM) [4]. The NAFV requires no parameter settings, has low computational costs, and effectively classifies players when their action frequencies are distinctly different, but is less effective when such differences are less apparent even though action orders are dissimilar. The HMM [5] is a powerful tool for classifying sequence data, but its performance depends on the structure and initial parameters.

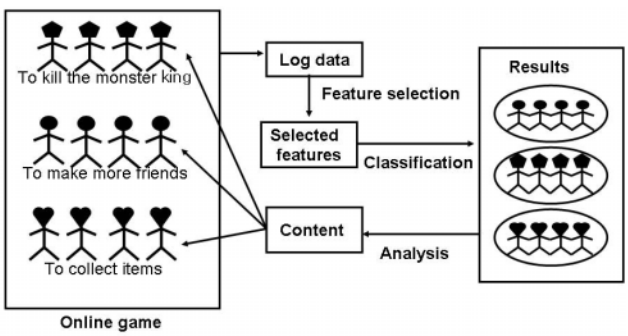

Fig. 1. Game Mining Concept.

In this paper, we propose a parameterless approach using action transition probability as the input feature to two classifiers and Kullback Leibler entropy [6] as the distance measure. The proposed approach considers action information in both frequency and order. In experiments, we compare it to the approaches mentioned above.

2. Player Modeling

We use Bartle's player categorization [7] as a foundation for modeling players. This categorization uses four types of player, i.e., Achievers, Explorers, Socializers, and Killers. Each player type has its own specific characteristics, motivations, and goals summarized as follows:

- **Achievers** want to have high status in the game world as soon as possible, and their main concerns are experience points, health levels, items that they posses, and high social standing.
- **Explorers** want to learn interesting things about the virtual game world such as easy ways to get valuable points, tricks to escape predators, and shortcuts to

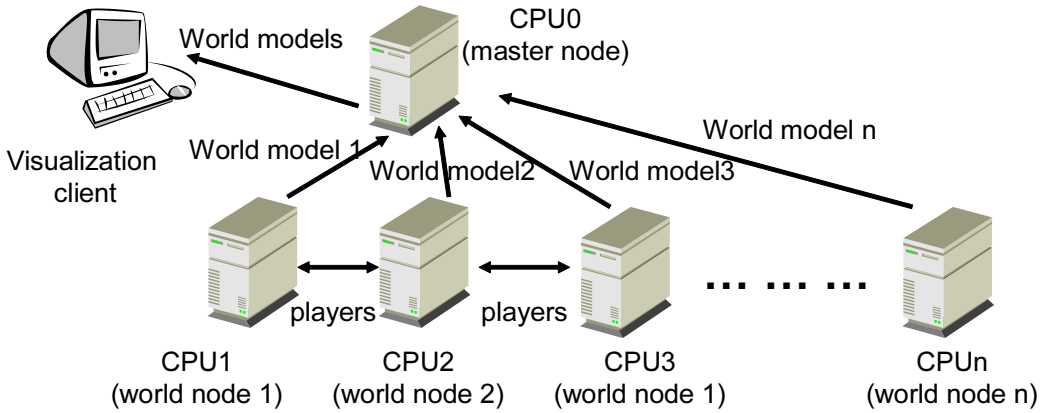

Fig. 2. Zereal architecture.

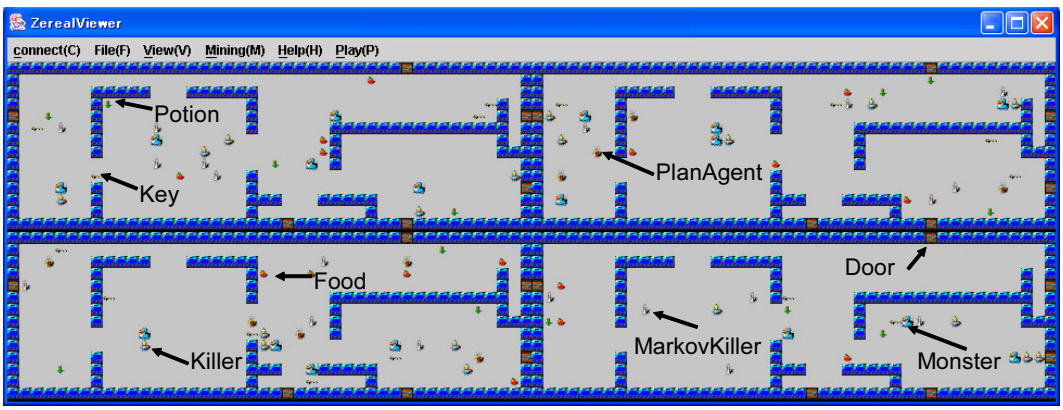

Fig. 3. Zereal's four game worlds.

arrive at destinations. They also enjoy demonstrating their knowledge to other players.

- **Socializers** are interested in relationships and communications with other players. They like talking to each other and trading things in the virtual game world.
- **Killers** want to kill players and monsters with the tools provided by the game, gaining satisfaction from their high fighting skills and the pain they inflict on others.

Below we describe the simulators, used in our experiments, for generating game logs on players of different types. A training data set consists of these logs and player labels. For simulation data, player types in training data are player models in the corresponding simulator. In actual games, information from player questionnaires or from Game Masters is used to label the type of player in a given training dataset.

2.1. Zereal Simulator

Zereal [8] is a multiagent simulation system that simulates a massively multiplayer online game.

Zereal consists of a visualization client, a single master node, and multiple world nodes (sub-nodes) (**Fig. 2**). Each world node controls its own world environment simultaneously. The master node gathers individual status information from world nodes, and forwards it to the visualization client. The visualization client receives this information and produces log files in different formats, including those for analysis and display (**Fig. 3**).

Figure 4 shows typical game logs. The first, second, and third columns therein are the simulation-time step, the current world node, and the actual clock time, respectively. The fourth column shows the agent identifier number, the fifth agent action, the sixth and seventh coordinates in the game world before and after the action. The last column is the agent type. Actions and agent types are extracted from game logs to form action sequences (**Fig. 5**), in which each character stands for an action described below.

Zereal uses three types of player agents, i.e., a Killer, Markov Killer, and Plan Agent. These are implemented having different characteristics and intelligence levels, but nine common actions, i.e., Walk(w), Attack(a), PickFood(f), PickPotion(p), PickKey(k), Talk(t), LeaveWorld(l), EnterWorld(e), and Removed(r). The Markov

```
58||1||2003-12-8: 20:6:35||1000321||pickupkey||(137,124)||(137,123)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000338||walk||(131,131)||(132,130)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000339||walk||(38,3)||(39,4)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000341||walk||(57,138)||(58,138)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000347||walk||(32,124)||(33,125)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000349||walk||(84,127)||(84,128)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000351||walk||(53,99)||(54,98)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000354||walk||(17,136)||(18,135)||PlanAgent1
58||1||2003-12-8: 20:6:35||1000355||walk||(84,33)||(83,34)||Killer1
58||1||2003-12-8: 20:6:35||1000356||walk||(76,51)||(76,52)||Killer1
58||1||2003-12-8: 20:6:35||1000357||walk||(69,38)||(70,37)||Killer1
58||1||2003-12-8: 20:6:35||1000358||walk||(47,8)||(48,8)||Killer1
58||1||2003-12-8: 20:6:35||1000359||attack||(135,39)||(135,38)||Killer1
                                    .
                                    .
                                    .
281||1||2003-12-8: 20:11:6||1000147||walk||(121,29)||(122,30)||MarkovKiller1
281||1||2003-12-8: 20:11:6||1000148||walk||(41,23)||(40,22)||MarkovKiller1
281||1||2003-12-8: 20:11:6||1000150||walk||(136,132)||(137,133)||MarkovKiller1
281||1||2003-12-8: 20:11:6||1000151||walk||(36,52)||(35,51)||MarkovKiller1
281||1||2003-12-8: 20:11:6||1000154||walk||(69,33)||(70,32)||MarkovKiller1
281||1||2003-12-8: 20:11:6||1000158||walk||(69,21)||(70,22)||Monster
281||1||2003-12-8: 20:11:6||1000160||walk||(130,88)||(131,87)||Monster
281||1||2003-12-8: 20:11:6||1000170||walk||(54,137)||(53,136)||Monster
281||1||2003-12-8: 20:11:6||1000191||attack||(71,93)||(70,92)||Monster
281||1||2003-12-8: 20:11:6||1000197||attack||(46,120)||(46,121)||Monster
281||1||2003-12-8: 20:11:6||1000204||attack||(16,52)||(15,51)||Monster
```

Fig. 4. Typical game logs.

```
.                .                .
.                .                .
.                .                .
MarkovKiller||1000059||wwwtwwwwwwttwwwwwttwwttwwtwwwtwwtwffwttwtwwwwpww
MarkovKiller||1000060||twwwawwwaawaaaawawtttwtwwwwwpwwwwwwwwwwwwwww
MarkovKiller||1000061||aaawwaawfwwkwwwwwwwwaawwwwawawfwfwwwwwwwwwwf
.                .                .
Killer||1000459|| wwaaaawwwwwwwaaaaaawfawwwwwwwwwwwwwwwwffwwwwwaaawww
Killer||1000460|| wwwawkwaaaaaaaaaaaaaaawwwwwwwwwwwwwwfwwwwwaaaaaawwww
Killer||1000461||wwaaaaaaaaaaaaaaawaaaaaaaawwwwwwwwwwaaaawwwwwawaaaww
.                .                .
PlanAgent||1000259||wkwwwwwwlewwwwwaawaaaaawwwwwawaaaaaawwwawwwaaaww
PlanAgent||1000260||wwawawwawaawaaaaaaaaawaaaaaawwwaaaaaawaaaaawkwwwww
PlanAgent||1000261||wwkwwwwkwwwwwfwwwwtawwwwwwwwaaaaawaawwwdwwwwwww
.                .                .
.                .                .
.                .                .
```

Fig. 5. Typical action sequences.

Killer is implemented by a Markov model consisting of eight states, i.e., fight, talk, hunt, transit, go for power up, flee, bored, and transported; only certain actions out of the nine can be performed in a given state.

2.2. Simac Simulator

We developed a simulator called Simac for simulating player types and actions not available in Zereal. Based on a simple Markov model for agent modeling, we developed Simac to generate reliable action sequences with more agent types and actions using the Python language. The Simac simulator has nine different states to represent situations of a player agent in playing, i.e., starting, moving, fighting, getting, losing, finding, talking, trading, and dying, plus 51 actions. Each action is assigned to one of the nine states.

Through the Markov model (**Fig. 6**), Simac can simulate many types of players. For reference, the Markov model of Killer is described in Appendix A. Simac action sequence formats are the same as those of Zereal.

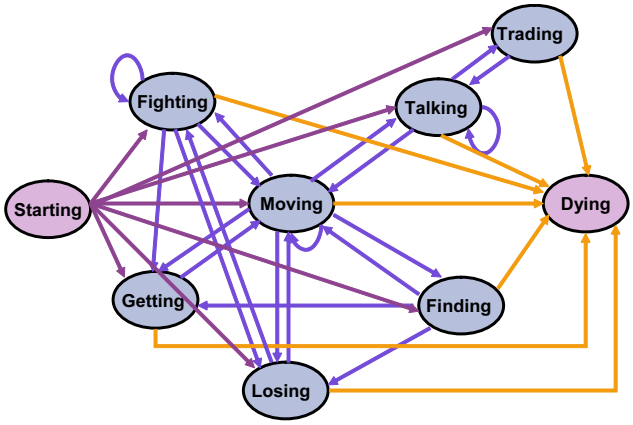

Fig. 6. Simac state diagram.

3. Methodologies

3.1. Action Transition Probability Matrix (ATPM)

The action transition probability matrix (ATPM) derived from action sequences takes into account action orders through action transition probabilities. It also holds action frequency information through its rows because the summation of each row is the action frequency of the corresponding action. The detailed procedure is shown as follows:

- **Step 1:** Initialize $m \times m$ action transition frequency matrix $\mathbf{F}$ by setting all elements to 1 where m is the number of actions. Thereby, only non-zero elements exist in the calculation of Kullback Leibler entropy described in the next subsection.

  Step 2: From left to right, scan subsequences of length two s_k, s_{k+1} in each action sequence $s = s_1, s_2, \ldots, s_l$, where l is the sequence length, and increment the element of matrix $\mathbf{F}$ that matches the transition of the current subsequence as follows:

$$\mathbf{F}_{qr} = \mathbf{F}_{qr} + 1 \quad \ldots\ldots\ldots\ldots \quad (1)$$

  where q and r are the indices of actions s_k and s_{k+1}, respectively, and k ($1 \leq k \leq l-1$) is the current scanning position.

- **Step 3 :** Calculate the **ATPM** by dividing each element of $\mathbf{F}$ with the summation of all elements as follows:

$$\mathbf{ATPM}_{ij} = \frac{\mathbf{F}_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{m}\mathbf{F}_{ij}}. \quad \ldots\ldots\ldots \quad (2)$$

3.2. Kullback Leibler Entropy (KLE)

Kullback Leibler entropy (KLE) [6] is one of the most widely used distance measures for comparing similarity between two probability distributions. Because KLE is asymmetric, we symmetrize it by defining distance $D(\mathbf{M}_a, \mathbf{M}_b) = D(\mathbf{M}_b, \mathbf{M}_a)$ as the average of two entropies

calculated from ATPMs of interest, $\mathbf{M}_a$ and $\mathbf{M}_b$, as follows:

$$D(\mathbf{M}_a, \mathbf{M}_b) = \frac{KLE(\mathbf{M}_a, \mathbf{M}_b) + KLE(\mathbf{M}_b, \mathbf{M}_a)}{2} \quad (3)$$

$$KLE(\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{u} \sum_{j=1}^{v} \mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{\mathbf{Y}_{ij}} \quad . \quad . \quad . \quad . \quad . \quad . \quad (4)$$

where $\mathbf{X}$ and $\mathbf{Y}$ have the size of $u \times v$.

4. Experiments

In our experiments, we used two classifiers with the ATPM and KLE, i.e., adaptive memory-based reasoning (AMBR) and prototype comparison (Prototype).

AMBR [9] is a variation of memory-based reasoning (MBR) [10]. MBR conducts majority voting of labels (or player class in our case) among the k nearest neighbors in the training dataset to classify a given player; parameter k must be decided in advance by the user. In contrast, AMBR is MBR with k initially set to 1; if ties in voting occur, it increments k accordingly until ties are broken.

In Prototype, the prototype of a particular class is determined by averaging the input features (either vectors or matrices) of all players of that class in the training dataset. To classify an unknown player, the player's input feature is compared to all prototypes to find the nearest one, and the player is classified in the class of the nearest prototype.

For comparing performance, we studied five classifiers in four datasets from [11], summarized as follows:

- **The ATPM AMBR** is the classifier using the ATPM as input feature to AMBR, where KLE is used for computing the distance between matrices.
- **The ATPM Prototype** is the classifier using the ATPM as input feature to Prototype, where KLE is used for computing the distance between matrices.
- **The NAFV AMBR** is the classifier using the NAFV as input feature to AMBR, where KLE is used for computing the distance between vectors.
- **The NAFV Prototype** is the classifier using the NAFV as input feature to Prototype, where KLE is used for computing the distance between vectors.
- **The HMM** is the classifier using a stochastic model to represent a prototype. For Zereal data, each prototype is trained by the Baum-Welch algorithm [5], where the number of states is set to the number of states in the simulator and initial parameters are set based on parameters in the simulator, as is done in [4]. For Simac data, each prototype is ideally constructed using the same structure and parameters as those in the simulator. For both Zereal and Simac data, the Viterbi algorithm [5] is used for calculating the similarity between the action sequence of a given player and each prototype.

The identification rate against trained data is used as the performance index of each classifier in all experimental trials, each trial using a different dataset.

4.1. Zereal Data with Three Types of Agent

Ten datasets were obtained from game logs generated by Zereal with three types of agent. In each dataset, 16 simultaneous game worlds were simulated for 300 simulation-time cycles with 50 Killers, 100 Markov Killers, 50 Plan Agents, 50 monsters, and 50 objects for each item type (food, potion, and key).

Agent types are summarized as follows:

- **The Killer** puts its highest priority on killing monsters.
- **The Markov Killer** gets as many items as possible to become stronger. Player agents of this type also attack monsters based on a predefined state-transition probability matrix.
- **The Plan Agent** combines the features of Killer, Achiever, and Explorer.

Results in **Fig. 7** show that the ATPM AMBR has the best performance, followed by the NAFV AMBR and HMM.

In this experiment, all player types have specific behavior patterns and clearly different action frequencies, except for Plan Agents that have combined behaviors and prototypes close to the input features of other-type players. As a result, both prototype classifiers have low performance. The performance of the HMM is lower than that of AMBR classifiers because the HMM is also a prototype-based classifier.

4.2. Zereal Data with Two Types of Agent

In the previous experiment, the NAFV AMBR has relatively good performance because most player types have clearly different action frequencies. The NAFV does not have information on action orders, however, and classifiers that use the NAFV as their input features do not perform well on data with similar action frequencies although they have different action orders. To verify this, we simulated two types of Markov Killer player, i.e., an InexperiencedMarkovKiller (IMK) and an ExperiencedMarkovKiller (EMK), by modifying the Markov model used in Zereal. They have similar action frequencies but different action orders. In each dataset, 16 simultaneous game worlds were simulated for 300 simulation-time cycles with 50 IMKs, 50 EMKs, 50 monsters, and 50 objects for each item type (food, potion, and key). These two agent types are summarized as follows:

- **The IMK** attempts all possible actions in a given situation equally, and all state transition probabilities in the Markov matrix are the same. This type represents novice players with little experience, acting randomly.

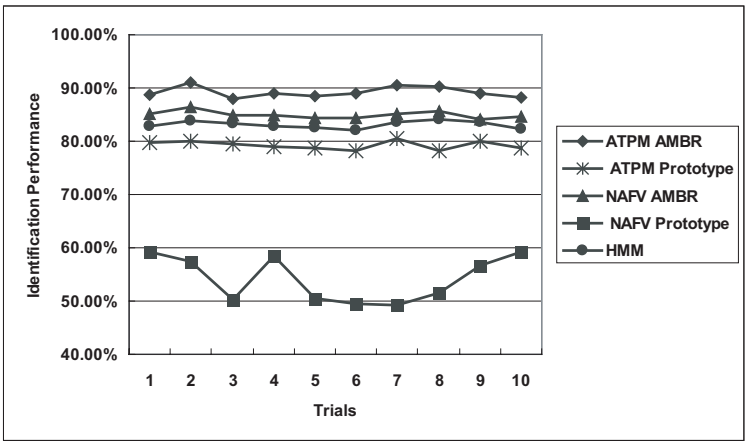

Fig. 7. Performance comparison for Zereal logs with three agent types.

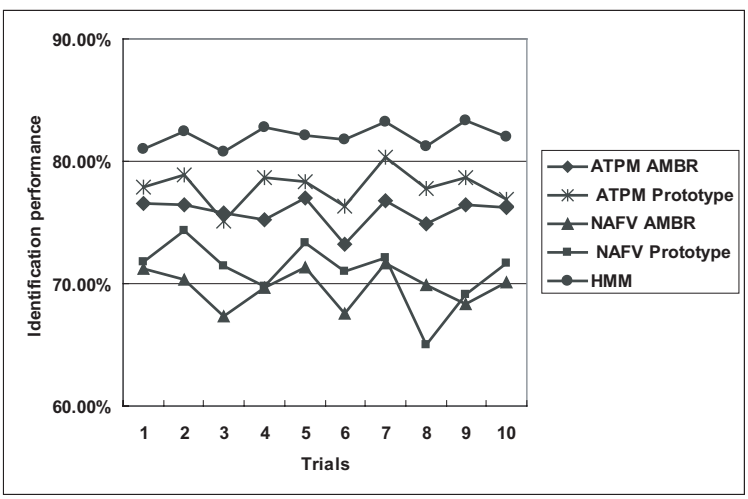

Fig. 8. Performance comparison for Zereal logs with two agent types.

- **The EMK** prioritizes some actions over others in a given situation, and state transition probabilities in the Markov matrix differ. This type represents veteran players with favored playing styles.

In the performances of the five classification approaches (**Fig. 8**), AMBR and Prototype using the NAFV, as their input features, are least effective, as expected. The HMM, which considers action orders, shows the best performance, followed by the ATPM Prototype and ATPM AMBR.

Although the HMM has the highest performance, it does not necessarily mean that the HMM is more pragmatic than the ATPM. This is because the HMM structure and initial parameters were set using a priori. It is commonly known that the HMM performance depends on its structure and initial parameters. To verify this, we trained the HMM with all parameters randomly initialized. The average performance for ten trials of the HMM with four, six, and eight states was 74.6% (standard deviation = 24.5), 82.1% (standard deviation = 27.5), and 78.2% (standard deviation = 27.8), respectively. In contrast, the ATPM Prototype and ATPM AMBR require no parameter settings.

4.3. Simac Data with Four Types of Agent

Each dataset was obtained by running 2000 agents for 500 simulation-time cycles. Four types of agent based on Bartle's categorization, i.e., Achievers, Explorers, Socializers, and Killers, were simulated (**Fig. 9**).

The ATPM Prototype has the best performance, followed by the HMM. For the same input features, the Prototype is superior to AMBR. Because the four player types implemented in Simac have distinct characteristics, prototype-based classifiers such as Prototype and the HMM are more appropriate than AMBR.

4.4. Simac Data with Two Types of Agent

In this experiment, two types of agents were implemented to obtain data in which each agent type has similar action frequencies but different action orders. Other

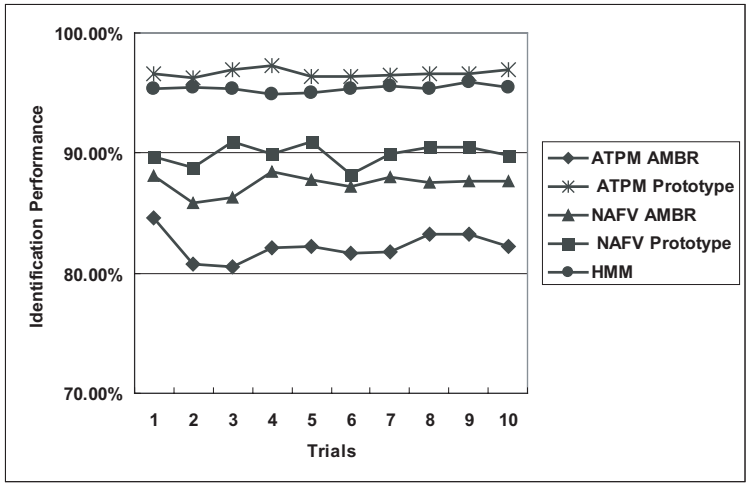

Fig. 9. Performance comparison for Simac logs with four agent types.

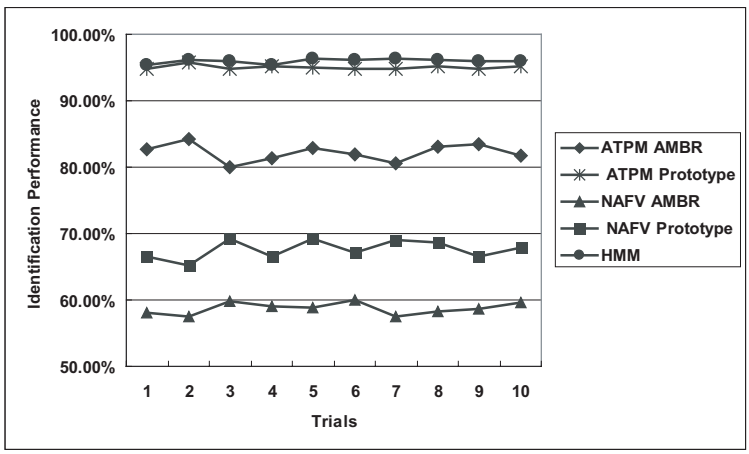

Fig. 10. Performance comparison for Simac logs with two agent types.

experiment settings were same as those in the previous subsection.

Agent types are summarized as follows:

- **The Killer** prioritizes killing, roaming around looking for monsters or players to attack. The difference between the Killer and the Strong Killer below is that the Killer requires more attacks to kill a monster because it is weaker than the Strong Killer.

- **The Strong Killer**, similar to the Killer, attacks anything alive. It kills monsters with fewer attacks than the Killer.

Killers and Strong Killers have typical action patterns with the same action frequencies but action orders are different reflecting their strength. A typical action sequence of the Killer is "wwwwaaaawwwwaaaaww" while that of the Strong Killer is "wwaawwaawwaawwaaww", with "w" and "a" standing for walk and attack, respectively.

The HMM has the best performance, followed by the ATPM Prototype with a shade of difference (**Fig. 10**). As explained earlier, the HMM's high performance is not surprising because knowledge in the simulator was used. The performances of both Prototype and AMBR using the NAFV as their input features are the lowest because the NAFV cannot handle action orders in data. Prototype performs better than AMBR.

5. Conclusions

We have proposed a new player classification approach (ATPM AMBR and ATPM Prototype) using the ATPM as the input feature to the two classifiers in use and KLE as the distance measure. The experimental results show that our approach has stable performance for player logs generated by Zereal and Simac. The ATPM Prototype performs better than the ATPM AMBR in identifying player types with distinct characteristics.

The existing approach using the NAFV (NAFV AMBR and NAFV Prototype) shows low performance except when player types have distinctly different action frequen-

Table 1. State transition probability matrix of Killers (Unit:%).

States	Moving	Fighting	Getting	Losing	Talking	Trading	Finding	Dying
Moving	40	19.99	10	10	10	0	10	0.01
Fighting	20	29.98	25	25	0	0	0	0.02
Getting	99.9	0	0	0	0	0	0	0.01
Losing	49.99	49.99	0	0	0	0	0	0.02
Talking	20	0	0	0	69.99	10	0	0.01
Trading	30	0	0	0	69.99	0	0	0.01
Finding	40	0	40	19.99	0	0	0	0.01
Dying	0	0	0	0	0	0	0	0

cies. In all cases, the HMM, with the use of a prior, shows comparatively high performance, especially for data with specific patterns of action orders. The HMM performance depends, however, on initial settings.

The proposed approach requires no parameter settings and is promising for online-game player classification. As our future work, we will conduct experiments with actual online game data and user behavior data in ubiquitous environments.

Acknowledgements

Work by the first author was supported in part by the Ritsumeikan University's Kyoto Art and Entertainment Innovation Research, a project of the 21st Century Center of Excellence Program funded by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) and by Grant-in-Aid for Scientific Research (C), No. 16500091, the Japan Society for Promotion of Science. The second author was supported in part by a MEXT scholarship. We thank the reviewers for their invaluable comments.

References:

[1] K. Alexander, R. Batle, E. Castronova, G. Costikyan, J. Hayter, T. Kurz, D. Manachi, and J. Smith, The Themis Report 2004 – Preview, 2004.

[2] R. Thawonmas, J. Y. Ho, and Y. Matsumoto, "User Type Identification in Virtual Worlds," Agent-Based Modeling Meets Gaming Simulation (Post-Proceedings of the Session Conference of the ISAGA, International Simulation and Gaming Association, 2003), Series: Springer Series on Agent Based Social Systems, Vol.2, K. Arai, H. Deguchi, and H. Matsui (Eds.), Springer, pp. 79-88, March, 2006.

[3] L. Shi and W. Huang, "Apply Social Network Analysis and Data Mining to Dynamic Task Synthesis for Persistent MMORPG Virtual World," Lecture Notes in Computer Science, M. Rauterberg (Ed.), Vol.3166 (Proc. ICEC 2004), pp. 204-215, Sep., 2004.

[4] Y. Matsumoto and R. Thawonmas, "MMOG Player Classification Using Hidden Markov Models," Lecture Notes in Computer Science, M. Rauterberg (Ed.), Vol.3166 (Proc. ICEC 2004), pp. 429-434, Sep., 2004.

[5] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. IEEE, Vol.77(2), pp. 257-285, Feb., 1989.

[6] G. Deco and D. Obradovic, "An Information-Theoretic Approach to Neural Computing," Springer, Berlin, Germany, 1996.

[7] R. Bartle, "Hearts, Clubs, Diamonds, Spades: Players Who Suit MUDs," The Journal of Virtual Environments, 1(1), 1996,

[8] A. Tveit, Y. Rein, V. I. Jorgen, and M. Matskin, "Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games," Proc. the 8th Scandinavian Conference on Artificial Intelligence (SCAI2003), Bergen, Norway, Nov., 2003.

[9] J. Y. Ho and R. Thawonmas, "Episode Detection with Vector Space Model in Agent Behavior Sequences of MMOGs," Proc. Future Business Technology Conference 2004 (FUBUTEC'2004), INSEAD, Fontainebleau, France, pp. 47-54, Mar., 2004.

[10] J. A. B. Michael and G. Linoff, "Data Mining Techniques: For Marketing, Sales, and Customer Support," John Wiley & Sons, Inc., N.Y., 1997.

[11] http://www.ice.ci.ritsumei.ac.jp/˜ruck/downloads.html

Appendix A. Markov Model of Simac Killer

Here, due to space limit, we only show the state transition probability matrix, the observation probability matrix, and the initial state distribution matrix of the Simac Killer in **Tables 1**, **2**, and **3**, respectively. The transition probability matrix of each agent type differs reflecting its specific characteristics. All agent types, however, have the same observation probability matrix and initial state distribution matrix.

Table 2. Observation probability matrix for all agent types.

State	Action
Moving	Walk (70%), Run (20%), Jump (10%)
Fighting	Use magic 1 (10%), 2 (10%), 3 (5%), Use weapon 1 (10%), 2 (10%), 3 (5%), Hit with hands strong (10%), middle (10%), weak (5%), Hit with legs strong (10%), middle (10%), weak (5%)
Getting	Get item 1 (10%), 2 (10%), 3 (10%), Get weapon 1 (10%), 2 (10%), 3 (10%), Get magic 1 (10%), 2 (10%), 3 (10%), Get power points (5%),Get life points (5%)
Losing	Lose item 1 (10%), 2 (10%), 3 (10%), Lose weapon 1 (10%), 2 (10%), 3 (10%), Lose magic 1 (10%), 2 (10%), 3 (10%), Lose power points (5%), Lose life points (5%)
Finding	Find secrete place 1 (20%), 2 (15%), 3 (15%), Find secrete item 1 (20%), 2 (15%), 3 (15%)
Talking	Talk (100%)
Trading	Sell item 1 (20%), 2 (15%), 3 (15%), Buy item 1 (20%), 2 (15%), 3 (15%)
Dying	Die (100%)

Table 3. Initial state distribution matrix for all agent types (Unit:%).

States	Moving	Fighting	Getting	Losing	Talking	Trading	Finding	Dying
Probability	99.95	0.01	0.01	0.01	0.01	0	0.01	0

Name:
Ruck Thawonmas

Affiliation:
Intelligent Computer Entertainment Laboratory, Department of Human and Computer Intelligence, Ritsumeikan University

Address:
1-1-1 Nojihigashi, Kusatsu, Shiga 525-8577, Japan

Brief Biographical History:
2002.4 Joined Ritsumeikan University as associate professor
2004.4 Promoted to full professor

Main Works:
• R. Thawonmas and S. Abe, "A Novel Approach to Feature Selection Based on Analysis of Class Regions," IEEE Trans. Systems, Man, and Cybernetics, Vol.27, Part B, No.2, pp. 196-207, 1997.
• R. Thawonmas, A. Cichocki, and S. Amari, "A Cascade Neural Network for Blind Signal Extraction without Spurious Equilibria," IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E81-A, No.9, pp. 1833-1846, 1998.

Membership in Learned Societies:
• IEEE Senior Member
• Association for Computing Machinery (ACM)
• The Institute of Electronics, Information and Communication Engineers (IEICE)
• Information Processing Society of Japan (IPSJ)
• International Game Developers Association (IGDA)

Name:
Ji-Young Ho

Affiliation:
Mobile R&D Group 1 (S/W Lab), Mobile Communication Division, Telecommunication Network Business, SAMSUNG ELECTRONICS CO.,LTD, Korea

Brief Biographical History:
2002.10-2004.9 Master's student at the Graduate School of Science and Engineering, Ritsumeikan University

Main Works:
• J.-Y. Ho and R. Thawonmas, "Episode Detection with Vector Space Model in Agent Behavior Sequences of MMOGs," Proc. of Future Business Technology Conference 2004 (FUBUTEC'2004), INSEAD, Fontainebleau, France, pp. 47-54, Mar., 2004.