Proc. 6th International Conference on Computer Games: Artificial Intelligence and Mobile Systems (CGAIMS ' 2005), Louisville, Kentucky, USA pp. 95-101, Jul. 2005.

HIDDEN MARKOV MODELS WITH FEATURE MAPPING: AN APPLICATION TO MMOG PLAYER CLASSIFICATION

Ruck Thawonmas and Yoshitaka Matsumoto
Intelligent Computer Entertainment Laboratory
Department of Human and Computer Intellgence, Ritsumeikan University
Kusatsu, Shiga 525-8577, Japan
URL: www.ice.ci.ritsumei.ac.jp
E-mail: ruck@ci.ritsumei.ac.jp

KEYWORDS

Hidden Markov Model, N-Gram, Term Frequency, CRM, MMOG

ABSTRACT

In this paper, we propose a method for determining the model structure and the initial parameters of Hidden Markov Models (HMMs) used for classification of players in Massively Multiplayer Online Games (MMOGs). The concept of the proposed method is that of mapping important features of each player type to its HMM states. Such important features are extracted by N-Gram and a variant of Term Frequency, techniques in Natural Language Processing. In our previous work, we discussed the use of HMMs for the aforementioned classification task. However, in the experiments there the model structure and the initial parameters of HMMs were determined using a priori knowledge on the player models of the MMOG simulator in use. With the proposed method in the present paper, the resulting HMMs have recognition performance higher than those initialized either by random setting of parameters or by assigning an equivalent value to all parameters in a same parameter set. In addition, they are comparable to those using a prior knowledge.

INTRODUCTION

The market size of Massively Multiplayer Online Games (MMOGs) continues to grow at a high speed. At the same time, competitions among MMOGs are also becoming very high. To keep players in the game, it is very important that players' demands are grasped, and that appropriate contents tailored for each player or each specific group of players are made.

In virtual worlds such as MMOGs, players are typically identified by their characteristics as "Killers", "Achievers", "Explorers", and "Socialisers" (Bartle 1996). Following this categorization, a typical implementation of customer relationship management (CRM) for MMOGs can be depicted in Fig. 1. In this figure, players are categorized into one of the pre-defined types based on appropriate selected features from game logs and are provided contents according to their favorites. Examples of such contents include those with more hunting opportunities, a wider variety of collectable items, longer-journey missions, and higher frequency of social events for Killers, Achievers, Explorers, and Socialisers, respectively. Thereby, players should enjoy the game more and hence play longer.

We previously showed in a paper (Matsumoto and Thawonmas 2004) that Hidden Markov Models (HMMs) (Bengio 1999) trained with player action sequences are effective in classification of MMOG players. In addition, they have higher recognition performance than a variant of k-nn classifier (Ho

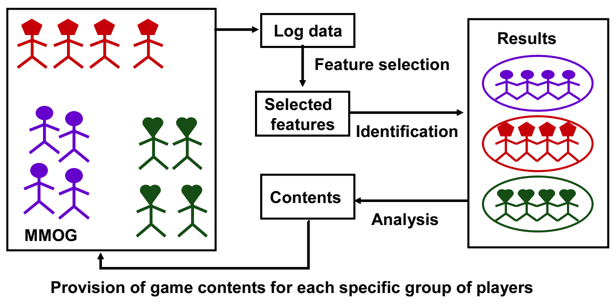


Figure 1: Typical Implementation of CRM for MMOGs

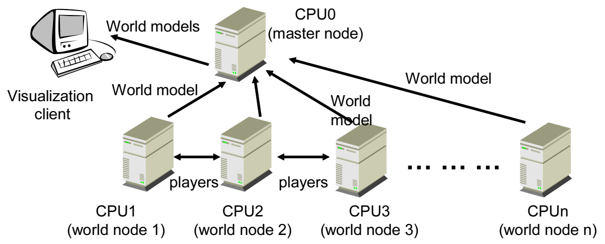


Figure 2: Architecture of the MMOG simulator

et al. 2003) trained with player action frequencies. However, the performance of HMMs is in general dependent on their model structure and initial parameters. In our previous paper (Matsumoto and Thawonmas 2004), we thus used a priori knowledge on the player models of the MMOG simulator in use. In practice, however, it is hard to obtain such knowledge in advance.

In this paper, we propose a method for determining the model structure and the initial parameters of HMMs by mapping important features of each player type to its HMM states. Such important features are extracted by N-Gram and a variant of Term Frequency, techniques in Natural Language Processing (Dale et al. 2000).

The rest of the paper is organized as follows. In the next section, we describe the MMOG simulator and the two types of player models that we use. Then, we describe a brief definition of HMMs and propose a method for mapping important features to HMMs. Next we present experimental results and end the paper with our conclusions and future work.

MMOG SIMULATOR

In general, it is difficult to obtain real MMOG game logs from game companies. So we use Zereal (Tveit et al. 2003), a Python-based multiple agent simulation system running on a PC cluster system. The architecture of Zereal and a screen shot of a game world are shown in Figs. 2 and 3, respectively. Zereal is composed of one master node and multiple world nodes. The master node collects the current status (world model) of each word and forwards this information to a client computer for visualization or data analysis. A world node simulates all objects such as player agents and monster agents. Other objects include food items and potion items for recovering stamina, and key items for opening a door in order to leave the current world.

In this work, we focus on two types of player agents, Killer and MarkovKiller because they somehow behave like Killers and Achievers, respectively. Each player agent has nine actions, i.e., 'walk', 'attack', 'chat', 'pickuppotion', 'pickupfood', 'pickupkey', 'leaveworld', 'enterworld', and 'removed'. The action 'removed' is outputted to game logs when a player agent (or a monster) dies due to its hit point having reached zero. Killer and MarkovKiller have different characteristics, as described below.

- **Killer (K)** has no sociability and will pursue the closest player or monster and kill it.
- **MarkovKiller (MK)** selects the next action from multiple candidates using a given Markov matrix. By manipulating the Markov matrix, we implement two types of MarkovKiller, InexperiencedMarkovKiller and ExperiencedMarkovKiller, described as follows:
  - **InexperiencedMarkovKiller (IMK)** who equally attempts all possible actions in a given situation; all elements of the Markov matrix equal.
  - **ExperiencedMarkovKiller (EMK)** who prefers particular actions over others in a given situation; the elements of the Markov matrix are not uniform.

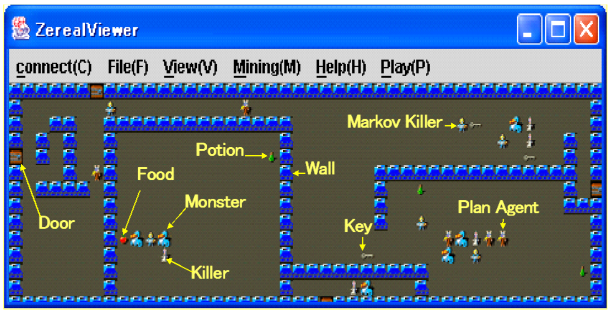


Figure 3: Screenshot of a game world

HIDDEN MARKOV MODELS

HMMs are a tool to statistically model a process that varies in time. From the set of observations, time structures hidden therein are derived. An HMM can be specified by (1) the set of hidden states $S = \{s_1, \ldots, s_H\}$; (2) the transition matrix A whose elements a_{ij} represent the probability to go from state s_i to the state s_j; (3) the set of observation symbols $V = \{v_1, \ldots, v_O\}$; (4) the emission matrix B whose elements b_{jk} indicate the probability of emission of symbol v_k when the system state is s_j; (5) the set of initial state probability distribution $\Pi = \{\pi_1, \ldots, \pi_H\}$ whose elements π_i represent the probability for s_i to be the initial state.

In our work, the Baum-Welch algorithm (Bengio 1999) is used to train HMMs, one for each player type, with a training data set. The training data set consists of both the action sequence and the type of each known agent player[1]. To identify the type of an unknown player agent, the Viterbi algorithm (Bengio 1999) is used. Inputted by the action sequence of an unknown player, this algorithm computes the log probability of each trained HMM. The unknown player agent will be labeled by the type of the trained HMM with the highest log probability.

[1]In practice, information from player questionnaires or from Game Masters could be used for labeling the type of a player in a given training data set.

FEATURE MAPPING

To determine the model structure and the initial parameters of HMMs, we use two algorithms, N-Gram and a variant of Term Frequency (TF). The basic idea behind our method is that of constructing and initializing HMMs based on important features extracted from the training data by these two algorithms. For player type t, we define an important feature as an N-gram word w with $TF_{w,t}$ beyond a given threshold ρ_t^{TF} and $NTF_{w,t}$ beyond a given threshold ρ_t^{NTF}, where $TF_{w,t}$ is the number of occurrences of w in t, and $NTF_{w,t}$ is calculated as follows:

$$NTF_{w,t} = \frac{TF_{w,t}}{\sum_t TF_{w,t}} \tag{1}$$

To limit the complexity of resulting HMMs, features that include those with smaller N, extracted previously, are discarded. In addition, for $N \geq 2$ a word is defined by symbols and their frequency, regardless of the order of them, in the word; for example, "abb" and "bab" are considered the same word. For each player type, an important feature is mapped to a state of the corresponding HMM, and the frequency of a symbol in the feature is used directly as the initial probability of emission of the symbol from the state.

The algorithm of the proposed method is given below as follows:

Step 1 - Initialize ρ_t^{TF} and ρ_t^{NTF}, and set $N = 1$.

Step 2 - Run N-Gram and flag N-gram words with $TF_{w,t} < \rho_t^{TF}$ for each player type t.

Step 3 - For each player type t, select from the unflagged N-gram words those that satisfy $NTF_{w,t} \geq \rho_t^{NTF}$ and do not include any smaller-size words selected in the previous iterations.

Step 4 - If no N-gram word has been selected for each player type in the current iteration go to **Step 5**; otherwise, increment N and go to **Step 2**.

Step 5 - For each player type t, construct the corresponding HMM such that it consists of states, individually mapped from the corresponding selected N-gram word, and an extra state mapped from the word consisting of all symbols.

Step 6 - Let H_{max} denote the number of states of the HMM with the maximum number of states. Add extra states to HMMs with the number of states less than H_{max} until its number of states becomes H_{max}.

Step 7 - For each player type t, initialize each element in the emission matrix B to the frequency of the corresponding symbol in the word mapped to the corresponding state, except for the extra states. The initial probability of emission of each symbol from the extra states is decided randomly. The initial value of each element in the transmission matrix A and in the set of initial state probability distribution Π is set to $1/H_{max}$.

Below we explain the above algorithm with simplified MMOG game log data shown in Figure 4. At **Step 1** and henceforth, ρ_t^{TF} and ρ_t^{NTF} are set, respectively, to $the\ number\ of\ \ type\ t\ players/2$ and $1.5/the\ number\ of\ player\ types$. In this example, thereby, $\rho_t^{TF} = 1.5$ and $\rho_t^{NTF} = 0.5$ for all t. Figure 5 shows the result of **Step 2**, where the elements with an oblique line are those with $TF_{w,t} < \rho_t^{TF}$. Figure 6 shows the result of **Step 3**, where the word "c" is selected for type 3. At **Step 4**, N is incremented to 2. Figure 7 shows the result of **Step 2** with $N = 2$, where the elements with an oblique line are those with $TF_{w,t} < \rho_t^{TF}$. Figure 8 shows the result of **Step 3** with $N = 2$; where the words "aa" and "ab" are selected for type 1, and the word "cc" is selected for type 2, and the word "bb" for type 3, and the elements with an oblique line are those that satisfy $NTF_{w,t} \geq \rho_t^{NTF}$, but include at least one of the previously selected words. At **Step 4**, N is incremented to 3. Figure 9 shows the result of **Step 2** with $N = 3$; where only words that occur in at least one player type are shown, and the elements with an oblique line are those with $TF_{w,t} < \rho_t^{TF}$. Figure 10 shows the result of **Step 3** with $N = 3$, where no word has been selected. Fig 11 shows the list of important features

Type 1	Agent 1	ababab
	Agent 2	abcaabaa
	Agent 3	d
Type 2	Agent 1	baccb
	Agent 2	bacabaab
	Agent 3	cc
Type 3	Agent 1	bcbaac
	Agent 2	cacabbcaca
	Agent 3	dbb

Figure 4: Player action sequences in simplified MMOG game log data

for each player type. The results of **Steps 5 − 7** are shown in Fig. 12, where a node represents a state, and x and y in $x : y$ next to a dotted arrow from a state represent a symbol and its emission probability from the state, respectively.

N=1	a	b	c	d
Type 1	8	5	~~1~~	~~1~~
Type 2	5	5	5	~~0~~
Type 3	6	6	6	~~1~~
Total	19	16	12	2

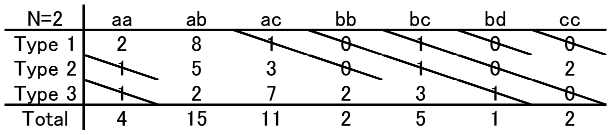
Figure 5: Result of **Step 2** with $N = 1$ for the data in Fig. 4

N=1	a	b	c	d
Type 1	0.42	0.31	0.08	0.50
Type 2	0.26	0.31	0.42	0.00
Type 3	0.32	0.38	**0.50**	0.50
Total	1.00	1.00	1.00	1.00

Figure 6: Result of **Step 3** with $N = 1$ for the data in Fig. 4

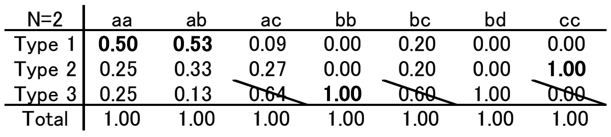
EXPERIMENTAL RESULTS

In our experiments, we generated game logs by running 10 independent Zereal games with 300 simulation-time cycles. Each game consisted of 16 worlds, in each of which there were 50 player agents of each type (**K** and **MK**), 50 monsters, and 50



N=2	aa	ab	ac	bb	bc	bd	cc
Type 1	2	8	~~1~~	~~0~~	~~1~~	~~0~~	~~0~~
Type 2	~~1~~	5	3	~~0~~	~~1~~	~~0~~	2
Type 3	~~1~~	2	7	2	3	~~1~~	~~0~~
Total	4	15	11	2	5	1	2

Figure 7: Result of **Step 2** with $N = 2$ for the data in Fig. 4



N=2	aa	ab	ac	bb	bc	bd	cc
Type 1	**0.50**	**0.53**	0.09	0.00	0.20	0.00	0.00
Type 2	0.25	0.33	0.27	0.00	0.20	0.00	**1.00**
Type 3	0.25	0.13	~~0.64~~	**1.00**	~~0.60~~	1.00	~~0.00~~
Total	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Figure 8: Result of **Step 3** with $N = 2$ for the data in Fig. 4

items for each game object (food, potion, and key). The total agent number of each type per game was thus 800. Next we transformed these raw game logs (as shown in Fig. 13) into sequences of actions (as shown in Fig. 14). Table 1 shows the average appearance frequency of each action, and the average length of the action sequences for each agent type.

We conducted experiments with four types of HMMs whose model structure and initial parameters are decided by

Random: initializing each element in B to a random value;

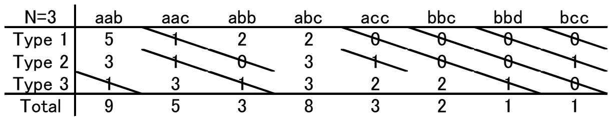
Equivalent: initializing each element in B to an equivalent value 1/9;

A Priori: initializing B with the matrix shown in Fig. 15, as done in the paper (Matsumoto and Thawonmas 2004); and

This Work: using the method proposed in the previous section.

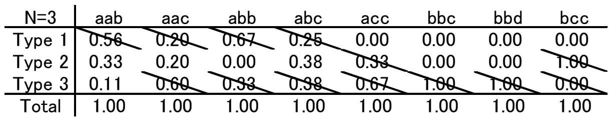
Table 1: Average appearance frequency of each action, and the average length of the action sequences for each agent type

	w	a	c	p	f	k	e	l	r	length
K	157.79	29.08	0.00	0.46	1.96	0.78	0.12	0.12	0.06	190
IMK	223.69	2.52	22.09	1.96	1.91	0.33	0.11	0.11	0.01	253
EMK	219.27	6.62	19.83	4.31	4.09	0.40	0.10	0.10	0.00	255



N=3	aab	aac	abb	abc	acc	bbc	bbd	bcc
Type 1	5	~~1~~	2	2	~~0~~	~~0~~	~~0~~	~~0~~
Type 2	3	~~1~~	~~0~~	3	~~1~~	~~0~~	~~0~~	~~1~~
Type 3	~~1~~	3	~~1~~	3	2	2	~~1~~	~~0~~
Total	9	5	3	8	3	2	1	1

Figure 9: Result of **Step 2** with $N = 3$ for the data in Fig. 4



N=3	aab	aac	abb	abc	acc	bbc	bbd	bcc
Type 1	~~0.56~~	~~0.20~~	~~0.67~~	~~0.25~~	0.00	0.00	0.00	0.00
Type 2	0.33	0.20	0.00	0.38	~~0.33~~	0.00	0.00	~~1.00~~
Type 3	0.11	~~0.60~~	~~0.33~~	~~0.38~~	~~0.67~~	~~1.00~~	~~1.00~~	~~0.00~~
Total	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Figure 10: Result of **Step 3** with $N = 3$ for the data in Fig. 4

For the first three HMMs, following the same recipe in the paper (Matsumoto and Thawonmas 2004), the number of states was set to 8 (cf. Fig. 15), and each element in Π and A was initialized to an equivalent value 1/8. Thus, some kind of a priori was also used in both Random and Equivalent.

The recognition rate of an HMM for each game was derived by averaging the recognition rates when the log data from worlds 1-8 and worlds 9-16 were respectively used for training and testing the HMM (whether or not it correctly labels an unknown data to either **K** or **MK**), and vice versa.

Figure 16 shows the recognition rates of the HMMs over 10 games. With the proposed method, the resulting HMMs have higher recognition performance than Random and Equivalent, and are comparable to A Priori.

CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a method for determining the model structure and the initial parameters of HMMs for classification of players in MMOGs. The resulting HMMs have higher recognition performance than other general approaches, such as random setting of parameters, or assigning an equivalent value to all parameters in a same parameter set. They also have the recognition performance comparable to the approach using a prior

	List of Words	
Type 1	ab	aa
Type 2	cc	
Type 3	c	bb

Figure 11: List of important features for each player type for the data in Fig. 4

knowledge, as adopted in our previous work. The proposed method requires decision of two parameters, i.e., ρ_t^{TF} and ρ_t^{NTF}. In this paper, we heuristically set them to $the\ number\ of\ \ type\ t\ players/2$ and $1.5/the\ number\ of\ player\ types$, respectively, which seems to work well for the log data in use. As our future work, we plan to conduct an extensive study on the effect of these parameters and to apply our findings to real MMOG data from an edutainment multiplayer online game called The ICE (Thawonmas and Yagome 2004), under development at our research laboratory.

ACKNOWLEDGEMENTS

This work has been 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, Science and Technology; and by Grant-in-Aid for Scientific Research (C), Number 16500091, the Japan Society for Promotion of Science.

References

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

[2] Yoshitaka Matsumoto and Ruck Thawonmas, "MMOG Player Classification Using Hidden Markov Models", Proc. of the Third International Conference on Entertainment Computing (ICEC 2004) Sep. 2004 Eindhoven, The Netherlands, published in Lecture Notes in Computer Science, Matthias Rauterberg (Ed.), vol. 3166, pp. 429-434.

[3] Bengio Y., "Markovian models for sequential data", *Neural Computing Surveys*, Vol. 2, pp. 129-162, 1999.

[4] Ho, J.Y., Matsumoto, Y. and Thawonmas, R., "MMOG Player Identification: A Step toward CRM of MMOGs", *Proc. the 6th Pacific Rim International Workshop on Multi-Agents(PRIMA2003)*, Seoul, Korea, pp. 81-92, Nov. 2003.

[5] Dale R., et al. (Editors), Handbook of Natural Language Processing, Marcel Dekker, 2000.

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

[7] Thawonmas, R. and T. Yagome. 2004., "Application of the Artificial Society Approach to Multiplayer Online Games: A Case Study on Effects of a Robot Rental Mechanism." *Proc. of the 3rd International Conference on Application and Development of Computer Games (ADCOG 2004)* (HKSAR, Apr.), pp. 12–17.

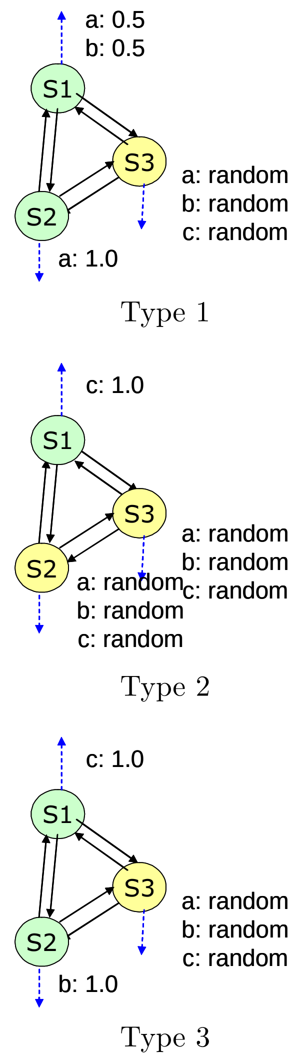


Figure 12: Resulting HMMs and their parameter initial values for the data in Fig. 4

```
1||1||2003-11-4: 19:6:35||1000085||walk||(73,43)||(74,43)||EMK
1||1||2003-11-4: 19:6:35||1000086||walk||(47,112)||(48,111)||IMK
1||1||2003-11-4: 19:6:35||1000297||pickuppotion||(106,135)||(106,136)||K
2||1||2003-11-4: 19:6:35||1000085||attack||(74,43)||(75,43)||EMK
2||1||2003-11-4: 19:6:35||1000086||walk||(48,111)||(48,112)||IMK
2||1||2003-11-4: 19:6:35||1000297||walk||(106,136)||(105,137)||K
```

Figure 13: Typical game log data from Zereal

```
K||wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwawwwwwwwwwwwwwwwwwwfwwwwww
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwfwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
wfwwwwwwwwwwwfwwwfwwwwwwwwwwwwwwwwwwwwwwfwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
wfwwwwwwwwwwfwwwwwwwwwwwwwwwwwwwwwwwwfwwwwwwwwwwwwwwwwwwfwwwwwww
K||waaaaaaaawwwaaaaaaaaawwwwwwwwawwawaaaaaaaaaaaaaaaawwwawwwawwwwwwwwwwwww
wwaaaawwwwwwwwwwwwwwwwaawwwwwwwwwwwwwwwwwwwwwwwwwwwffwwwwwwwwawwwwwwwwwwww
wwwwwwwwwwwwwawwwwwwwwwwwwwwawaaawawwwwwwwwwwwwwwwwwwwwwfwwwwwwwwwwwwww
wwwwwwwfwwwwwwfwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwfkwwww
IMK||wwwwwwwwwwwwwwwwwwwwwwwwawwawwwwtwawwwwtwwwwfwtwwwwwwwwwwwwwwwwwwwt
wwwwwwwwttwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwpwwwwwwwwwwwwwwwwwwwwwwwwwwwtwwwt
twtwwwwwwwwtwwwwwwwwwwwwwwwwwwwwwwwwwwwwtwwwwwwwwtwwpwwwwwwwttwwtwwwwwwwtww
wwwwwwwwwfwwwwwwwwwwwwwwwwwwwwwwwwwtwfwttwtwtwwwwtttttttwwwtwwwwttwwtwwwwwt
EMK||wwwwwwwaawwawaawwwwwwwwwwwwwwwwwawtawwawwtwwwwttwwwwwttwwwwwwwwwwww
wwwwwwwwwwwwwaaawwwwwttwwwwwwwwwwwwwtwwwwwwwwwwtwwwwwwwwttwwttwwwwwwttttwwttw
wwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwfwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwwww
wwwpwwwwwwwwwwwwkwwwwwwwwpwtwwwttwwwtwwwtttttwwwwwwwwwwwwwwwwwwpwwwwwwwwwwww
wwwwwwwwpwwwwfwwwwww
```

Figure 14: Typical player action sequences

	w	a	c	p	f	k	e	l	r
fight	0.00	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.25
talk	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
hunt	0.70	0.00	0.00	0.10	0.10	0.10	0.00	0.00	0.00
transit	0.75	0.00	0.00	0.00	0.00	0.25	0.00	0.00	0.00
go for powerup	0.00	0.00	0.00	0.50	0.50	0.00	0.00	0.00	0.00
flee	0.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25
bored	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
transported	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.50	0.00

Figure 15: Initial emission matrix having 8 states and 9 symbols (w: walk, a: attack, c: chat, p: pickuppotion, f: pickupfood, k: pickupkey, e: enterworld, l: leaveworld, r: removed) derived based on a priori knowledge on the Zereal player models

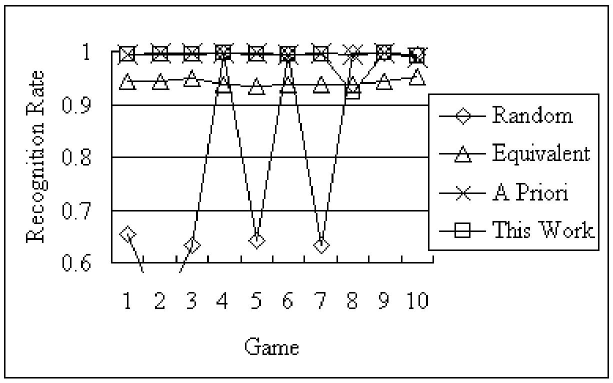


Figure 16: Recognition rates of four types of HMMs over 10 games