Game Bot Detection Approach Based on Behavior Analysis and Consideration of Various Play Styles

Yeounoh Chung, Chang-yong Park, Noo-ri Kim, Hana Cho, Taebok Yoon, Hunjoo Lee, and Jee-Hyong Lee

An approach for game bot detection in massively multiplayer online role-playing games (MMORPGs) based on the analysis of game playing behavior is proposed. Since MMORPGs are large-scale games, users can play in various ways. This variety in playing behavior makes it hard to detect game bots based on play behaviors. To cope with this problem, the proposed approach observes game playing behaviors of users and groups them by their behavioral similarities. Then, it develops a local bot detection model for each player group. Since the locally optimized models can more accurately detect game bots within each player group, the combination of those models brings about overall improvement. Behavioral features are selected and developed to accurately detect game bots with the low resolution data, considering common aspects of MMORPG playing. Through the experiment with the real data from a game currently in service, it is shown that the proposed local model approach yields more accurate results.

Keywords: User behavior analysis, machine learning, bot detection model, MMORPGs, game play styles.

Manuscript received Mar. 31, 2013; revised Aug. 26, 2013; accepted Sept. 6, 2013.

This work was supported by Ministry of Culture, Sports and Tourism (MCST) and Korea Creative Content Agency (KOCCA) in the Culture Technology (CT) and Research Development Program 2012.

Yeounoh Chung (phone: +82 31 290 7987, yeounohster@gmail.com), Chang-yong Park (codep@skku.edu), Noo-ri Kim (pd99j@skku.edu), Hana Cho (cho2405@skku.edu), and Jee-Hyong Lee (corresponding author, john@skku.edu) are with the School of Information and Communication Engineering, Sungkyunkwan University, Suwon, Rep. of Korea.

Taebok Yoon (tbyoon@seoil.ac.kr) is with the Department of Computer Software, Seoil University, Seoul, Rep. of Korea.

Hunjoo Lee (hjoo@etri.re.kr) is with the Creative Content Research Laboratory, ETRI, Daejeon, Rep. of Korea.

I. Introduction

Game bots are automated programs that perform repetitive tasks on behalf of human players. Since game bots can perform tedious tasks without breaks, they are usually used to obtain unfair advantages over honest human players. An unruly swarm of game bots depletes game contents and resources, so honest players may feel discouraged and lose interest and eventually retire from the game. Game bots are potentially serious threats to gaming businesses, and detecting game bots is of great importance to game publishers.

Much effort has been made on the game publisher side to prevent the use of game bots. However, it is hard to detect game bots because they simulate legitimate human game playing behavior. Various strategies have been proposed for game bot detection, including administering a Turing test, network traffic analysis, and bot scanning, but these strategies are known to have drawbacks, such as interfering with game play or being easily evaded by game bots [1]-[6].

A promising alternative to the traditional bot detection approaches is behavior analysis of game playing [7]. To this end, researchers analyze behavioral patterns of players. However, the previous work tried to discriminate human players from game bots with one global model. Players have various gaming styles; some users are battle-oriented, some are item-collection-oriented, and some are quest-oriented. Since such players exhibit different and distinct behavioral patterns, behavior-based bot detection should use a different set of bot detection rules for each type of player group. However, previous research did not consider the differences in behavioral patterns of player groups. Instead, a global bot detection model was used for the different player groups.

Another limitation of the previous work is that researchers only examined one or two characteristics of behavior for bot detection rather than all aspects of behavior [8]-[10]. Observing one or two types of behavior can still accurately detect game bots; however, because distinguishing patterns for bot detection vary greatly from game to game, the previous approaches can be highly game dependent. For instance, if detecting game bots based on party play (party duration) [11], then the detection mechanism works well only with game bots in parties and does not effectively work for players who do not party or share their collections. Thus, it is important for a generic bot detection methodology to consider whole aspects of behavior to cover game bots with different gaming styles.

Another concern for behavior-based bot detection approaches is the game server workload for collecting game play data of all the players. In general, behavior-based game bot detection approaches require game play data in high resolution in time; however, collecting such high resolution data for every player can overload the game servers in service. Although it is important to minimize the workload of the game servers in collecting game play data, most of the previous behavior-based bot detection approaches did not address this concern [12], [13].

In this paper, we propose a novel bot detection methodology based on behavior analysis. We group players by their behavioral similarities and detect game bots within each group, using a set of customized bot detection rules for each group of players. We overcome the limitation of using a global bot detection model for different player groups by using local models for player groups, grouped by behavioral similarities.

Additionally, we examine the behavioral features that can be extracted from the low resolution game data to reduce the game server workloads. The features are common in massively multiplayer online role-playing games (MMORPGs) and reflect all the aspects of playing behavior. Features reflecting all the aspects of playing behavior are necessary to effectively detect game bots with low resolution data, as well as to reduce the game dependency of the proposed method.

We evaluate the proposed method using a large dataset from an MMORPG currently serviced by a Korean game publisher. Our experiment results confirm the importance of considering various aspects of game behavior and different gaming styles to detect game bots.

The rest of this paper is organized as follows. Section II reviews related work in game bot detection. Section III describes the features used for our experiment and the proposed bot detection framework. Section IV presents and explains the experiment results. Section V summarizes and concludes the work.

II. Related Work

It is difficult to detect game bots because they simulate legitimate human game playing behavior. In an effort to restrain game bots in MMORPGs, various strategies have been proposed, such as repeatedly performing a Turing test, network traffic analysis, bot scanning, and behavior analysis.

Using a Turing test such as CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) provides a suitable way to detect game bots [1], [2]. The basic idea of a Turing test is to ask a question that is easy for humans to answer but very difficult for computers. Based on the answer from players, one judges if the respondent is human or not. The CAPTCHA authentication is widely used in MMORPGs for bot detection [3], [7]. However, Q&A-based approaches interfere with normal game play, and some advanced game bots have an evasion function that thwarts CAPTCHA authentication [14], [16].

Chen and others [17] traced game data packets for bot detection. They showed that there were differences in network traffic patterns generated by humans and game bots. Bot detection measures based on network traffic analysis are less obtrusive to human players than a Turing test or anti-bot software [9], but they can cause network overload and lags in game play [11].

Client-side bot scanning is one of the traditional anti-bot methodologies. Anti-bot software for bot scanning is installed on client computers and examines event sequences and memories. This can cause inconveniences for players, such as collisions with operating systems [11]. In many cases, such a bot detection mechanism can be evaded by running games in guest mode on an administrator account [13].

A promising alternative to the above approaches is bot detection based on player behavior analysis. To this end, researchers analyzed various behavioral patterns of game players using machine learning or statistical techniques.

Some researchers focused on movement pattern of game play behavior [18]-[21]. This exploits the fact that bots move along prescribed paths, as bots show regularities in their movement patterns, whereas human players follow more complex and random movement paths. However, bot writers can easily introduce some irregularities to bots' movement patterns [13].

Other features that are preferred for bot detection are attack sequences [8], [10] and social sequences [22]-[24], because bots usually have regular patterns in those features. However, they are also limited in that such features may not be common or relevant in game bots, depending on the game or gaming purpose.

One recent study analyzed party play behavior to detect game bots. The authors showed that game bots tend to stay in the party much longer (almost indefinitely) than human players [11]. This approach also has limitations; their bot detection mechanism solely depended on party play behavior, which may not work for game bots playing individually.

A limitation of the approaches mentioned above is that the researchers considered only a single aspect of player behavior. The features of a particular behavioral aspect may be very effective in some games but may not work effectively in some others to detect bots. Furthermore, even if the features of the particular behavioral aspect are effective, they may not be practically applicable if collecting game data for the features overloads the game servers in service. Hence, more generic and practically applicable features need to be observed.

Another limitation is that they did not consider the fact that players have quite different behavior patterns. The previous work tried to discriminate between human and bot players with one global model. Since there are various game playing styles, it is difficult for a single model to effectively handle all the players. The overall performance of a global model may be low and some specific types of players may not be correctly classified.

III. Proposed Method

In this section, we propose a bot detection approach based on player behavior analysis. The proposed bot detection approach works in two steps. First, players are clustered by their behavioral similarities. Second, a local model with customized bot detection rules is generated for each cluster. To describe the proposed methodology, we first explain the features selected and developed for low resolution data and then bot detection model construction considering different types of players.

1. Feature Selection

In general, behavior-based bot detection methods can yield accurate results because such methods exploit key behavioral patterns that are different in humans and game bots. However, collecting and monitoring game behavior data for every player can be very expensive. Therefore, behavioral features that are effective and less burdensome to the systems need to be carefully chosen and developed.

To reduce the game server workloads in data collection, we select simple behavioral features that are extractable from the game data collected at a low sampling rate. Some of the simple behavioral features are attack counts, hit counts, and item counts, which are either accumulated values during the sampling interval or snapshot values at the end of the sampling interval.

Table 1. Description of behavioral features.

Feature	Description
Hunting	Accumulated number of hunted NPCs for sampling interval
Attack	Accumulated attack count for sampling interval
Hit	Accumulated hit (successful attack) count for sampling interval
Defense	Accumulated defense count for five minutes
Avoidance	Accumulated avoidance (successful defense) count for sampling interval
Recovery	Accumulated number of healing portion usages for sampling interval
Item	Number of items at time of sampling
Collection	Accumulated number of collected items for sampling interval
Drop	Accumulated number of dropped items for sampling interval
X	Coordinate X at time of sampling
Y	Coordinate Y at time of sampling
Portal	Accumulated number of portal item usage count for a sampling interval

Since we assume a low resolution of the collected data, some of the informative behavioral features used in the previous research are not used in this study. For instance, repeated moving paths and attack sequences are suitable features for behavior-based bot detection, but those features require the game data sampled at a fine resolution.

The features are selected considering the three categories of game behavior: Battle, Collect, and Move. These categories are common in MMORPGs and thus can represent important aspects of game behavior and players with various play styles. A general bot detection approach should consider various aspects of playing behavior because distinguishing patterns for bot detection vary greatly from game to game. Furthermore, the data sampling resolution is very low, so the approach must consider various aspects of game playing behavior.

For the three categories of game behavior, we choose features that are very common in most MMORPGs. For Battle, we choose Hunting, Attack, Hit, Defense, Avoidance, and Recovery. For Collect, we choose Item, Collection, and Drop. Fore Move, we choose X, Y, and Portal. The features presented in Table 1 are designed based on these behaviors. So, these features are generally applicable and can be evaluated from game playing behaviors in most MMORPGs.

As mentioned, the selected features are simply accumulated or snapshot values. For example, Attack indicates how many times a player attacked within a sampling interval on average, but the feature does not convey any information on how effectively the player attacked. To grasp deeper meanings of

Table 2. Developed feature set.

Feature	Formula
Combat Ability I	$\frac{Hunting}{Attack}+\frac{Avoidance}{Defence}$
Combat Ability II	$\frac{Recovery}{Hunting}$
Combat Ability III	$\frac{Hunting}{\sqrt{\lvert X-pre\_X\rvert^2+\lvert Y-pre\_Y\rvert^2}}$
Collect Pattern	$\frac{(Collection+Drop)}{2\times Item}$
Move Pattern	$\sqrt{\lvert X-pre\_X\rvert^2+\lvert Y-pre\_Y\rvert^2}$

player actions, we develop new features based on the selected feature set: Combat Ability I, Combat Ability II, Combat Ability III, Collect Pattern, and Move Pattern. The formal definitions of the developed features are shown in Table 2.

These features are developed to measure the effectiveness and efficiency of player behavior. Combat Ability I measures how effectively a player attacks and defends, by considering the attack success rate and defense success rate. Combat Ability II measures how health-efficiently a character combats based on the number of healing portion usages per hunting (killing a monster). Using a lesser number of healing portions for every hunt indicates that the player manages his or her health efficiently in combat. Combat Ability III measures how space-efficiently a player battles based on the number of monsters hunted and the amount of distance traveled. Hunting more monsters within a shorter distance indicates that the player efficiently hunts monsters.

Collect Pattern measures how efficiently a player uses his or her item slots. Emptying full item slots and filling empty item slots represents efficient use of item slots. In addition, Move Pattern is evaluated to observe the range of movement within a fixed time period. This feature does not measure the effectiveness or efficiency of player behavior, but there can be a correlation between the range of movements and play styles. For example, quest-oriented players would travel more distance than players with other gaming purposes.

Both the selected and the developed features are standardized with Z-scores to equally weight each behavioral feature. This step is of particular importance because features have different value ranges.

2. Player Grouping and Bot Detection Model Generation

Game players have different gaming purposes and behavioral patterns, which results in various game play styles.

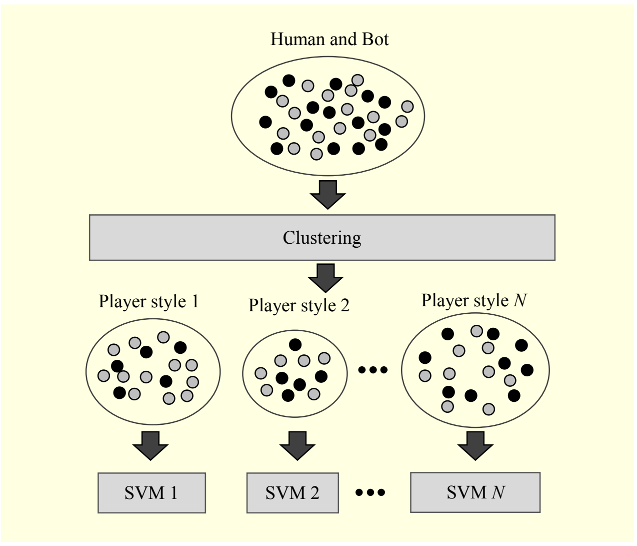

Fig. 1. Bot detection considering different game play styles.

So, it is necessary in bot detection based on behavior analysis to consider various game play styles and use a different set of bot detection rules or a local model for each play style.

In this paper, we propose a bot detection methodology, concerning various game play styles in MMORPGs. We first examine three aspects (Battle, Collect, and Move) of player behavior that are relevant to common MMORPG play and then group players by their behavioral similarities based on player behavior. Players who have similar gaming purposes and preferences may also exhibit similar behavioral patterns. Hence, we can group players by their game play styles using their behavioral similarities.

Figure 1 gives an overview of the proposed methodology. We use a *k*-means algorithm for grouping players by their behavioral similarities. The *k*-means algorithm is a general clustering technique for grouping data into *k* clusters, for a given *k*. Given player data with behavioral features, the *k*-means algorithm groups together players who have similar behavior patterns by maximizing intra-cluster behavioral similarities and minimizing inter-cluster behavioral similarities. Therefore, we can obtain *k* player groups, in which players have similar game playing patterns.

For each cluster of similar players, a support vector machine (SVM) bot detection model is generated. The local SVM model is trained with human and bot player data from the associated cluster. An SVM is a well-known machine learning technique used in binary classification. SVMs require only a few parameters for model tuning, and the method is well known for its low generalization error [25]. Since the proposed method develops local models for each group, local models can be best customized to detect game bots in each player group. Such local bot detection models would more accurately

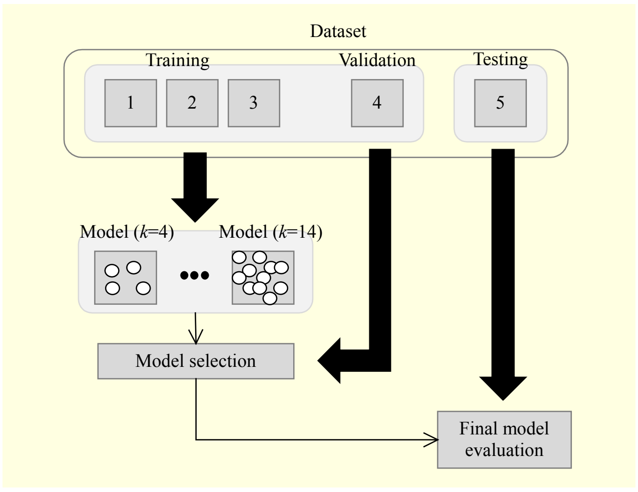

Fig. 2. Process for finding proper number of clusters, k.

and effectively detect game bots within each player group. The combination of these models can outperform an approach based on a single global model.

If an unknown player is given, the player is classified as a human player or a game bot, using the generated k clusters and the k SVM models. First, the most similar cluster to the player is identified. Euclidean distances between the player behavioral feature vector and cluster centroids are used as a similarity measure between players and clusters. If the player belongs to cluster i, then its local SVM model, SVM_i, is used. The behavioral feature vector of the player is input into the SVM_i, and the player is classified as the output of SVM_i: a human player or a game bot.

Each player group may consist of both human and bot players with similar behaviors. Therefore, a local bot detection model for a particular player group can focus more on the behavioral differences between human players and bot players and less on the behavioral differences among different play styles, which may act as noises in using a global bot detection model. Hence, we can build local detection models that are well adapted to each group with higher accuracy.

However, if the number of groups is too small, players with less behavioral similarities can be clustered together. If the number of groups is too large, each cluster may contain too few players to build a bot detection model for the cluster. In such cases, the proposed method may be inaccurate. Hence, the number of clusters, k, should be carefully chosen. In this study, we try four to 14 clusters and select the number of clusters that showed the best performance using validation data.

Figure 2 describes the process for finding the proper number of clusters, k. The dataset is evenly and randomly divided into five folds. Then, four folds are used for model building, and the remaining one fold is used for evaluating the performance of the final model. At the model building step, we first build 11 models with three training folds. Each model has a different number of clusters from four to 14. Next, each model is evaluated using the validation data. The model that produces the maximum performance is selected as the final model. The final model is the output of the model building step. Finally, the performance of the final model is evaluated with the testing data, which is independent of the data for model building. Throughout the experiment, we use five-fold cross-validation to reduce statistical variance.

IV. Experiment

1. Raw Data

We collect data from the popular Korean MMORPG game *Yulgang Online*. The game has more than 200 million subscribers in total, with a maximum of 85,000 concurrent connections.

Most game publishers strictly prohibit the use of game bots, but *Yulgang Online* allows players to purchase and use a game bot item called "the Box of Black Soul." Since the game bots also perform repetitive tasks with given programmed rules as illegal games bots do, we may use the data of *Yulgang Online* to develop a bot detection methodology.

Data is collected from a game server in service for a month. To minimize the workloads of the game server in service, the length of sampling intervals is set to five minutes, which is a long interval to obtain descriptive behavior data.

A brief summary of the collected behavior log data is shown in Table 3. In Table 3, the number of logs reflects the data collected at five-minute intervals. As shown, 20,543 human players create 3,529,099 logs. There are twice as many human players as game bot players, yet there are 3.6 times more bot player logs than human player logs. In MMORPGs, there are generally more human players than bot players, but there are more bot players in *Yulgang Online* because the use of game bots is allowed.

We extract bot player instances and human player instances at a one-to-one ratio. Since there are more bot players than human players, using all data instances would result in biased bot detection models. To generate more general unbiased bot detection models, we undersample bot player instances.

Table 3. Number of logs and characters.

Group	Number of logs	Number of players
Human player log	3,529,099	20,543
Bot player log	12,968,264	10,282
Summation	16,497,563	30,825

Finally, we standardize features of data instances with Z-scores to equally weight each behavioral feature. This step is of particular importance, because features have different value ranges. Equation 1 shows the Z-score standardization.

$$f_i^z = \frac{f_i - m_f}{\sigma_f} \tag{1}$$

In the equation, f_i is the value of feature f of instance i, and m_f and σ_f are the average and the standard deviation of feature f of all players, respectively. The Z-score of f_i is represented by f_i^z.

2. Baseline and Evaluation Metrics

To compare the performance of the proposed method, we use a global SVM bot detection model as the baseline, which does not consider game play styles. Most of the previous approaches were based on a single global model, so the model based on a single SVM is chosen as the baseline.

We use four evaluation metrics: *Accuracy*, *Precision*, *Recall*, and $F1$. *Accuracy* measures how many bots and humans are correctly identified. *Precision* measures how many of the players detected as bots are really bots. *Recall* measures how many of the real game bots are detected. More formal definitions are as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{2}$$

$$Precision = \frac{TP}{TP + FP}, \tag{3}$$

$$Recall = \frac{TP}{TP + FP}. \tag{4}$$

In the equations, *TP*, *TN*, *FP*, and *FN* are the number of real bots identified as bots, the number of humans identified as humans, the number of humans identified as bots, and the number of bots identified as humans, respectively.

Usually, *Precision* and *Recall* tend to be inversely proportional to each other. If *Precision* is high, *Recall* is low, and vice versa. So, we observe one more measure, $F1$, which combines *Precision* and *Recall* as follows:

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{5}$$

This is the harmonic mean of *Precision* and *Recall*.

3. Bot Detection Results

In this experiment, we show that the proposed method outperforms the baseline. For the experiment, we use six different feature sets for two reasons. First, it is to verify that the superiority of the proposed method over the baseline is independent of the feature sets. Second, it is to verify that the selected and developed feature sets are effective over single aspect feature sets.

The six feature sets used in the experiment are summarized in Table 4. F_{12} represents the 12 selected features in section III.1. F_5 represents the five developed features in section III.1. F_B, F_C, and F_M are subsets of F_{12}, categorized as Battle, Collect, and Move feature groups in section III.1, respectively. Lastly, F_{17} is the union of F_{12} and F_5. F_{17}, F_{12}, and F_5 are whole aspect feature sets, and F_B, F_C, and F_M are single aspect feature sets.

Table 5 shows the baseline results, and Table 6 shows the proposed method results. For any feature sets, the proposed approach gives superior results to the baseline. In *Accuracy*, the performance of the proposed method is higher than that of the

Table 4. Feature sets used in experiment.

	Feature	Description
Whole aspect	F_{12}	12 selected features in section III.1
	F_5	5 developed features in section III.1
	F_{17}	12 selected and 5 developed features ($F_{12} \cup F_5$)
Single aspect	F_B	Subset of F_{12}, containing only Battle aspect features
	F_M	Subset of F_{12}, containing only Move aspect features
	F_C	Subset of F_{12}, containing only Collect aspect features

Table 5. Baseline result.

(%)

	Feature	Accuracy	Precision	Recall	F1
Whole aspect	F_{17}	94.27	94.69	93.90	94.29
	F_{12}	84.72	81.42	87.17	84.20
	F_5	88.07	95.17	83.33	88.86
Single aspect	F_B	80.25	68.17	89.88	77.53
	F_M	59.43	63.22	58.77	60.91
	F_C	67.31	61.57	69.55	65.31

Table 6. Proposed method result.

(%)

	Feature	Accuracy	Precision	Recall	F1
Whole aspect	F_{17}	96.20	95.95	96.42	96.18
	F_{12}	91.50	91.68	91.35	91.51
	F_5	89.61	93.62	86.66	90.01
Single aspect	F_B	87.87	86.20	89.17	87.65
	F_M	72.38	71.88	72.61	72.24
	F_C	68.14	69.40	67.68	68.53

Table 7. Four player types in MMORPG from the Bartle test of gamer psychology [26].

Player type	Explanation
Killer	Players prefer fighting other players and engage in battles.
Achiever	Players who prefer to gain experiences (skill levels), items, and other game rewards.
Explorer	Players who find great joy in discovering areas, maps, and hidden places.
Socializer	Multitude of players who play games for the social aspect, rather than the actual game itself.

Table 8. Player type distribution.

	Killer	Achiever	Explorer	Remainder
Human players	777	1,266	224	6,186
Game bots	2,296	1,895	11	4,251
Total	3,073	3,161	235	10,437

baseline under any feature sets. In *Precision* and *Recall*, the baseline is a little higher in some cases. However, in *F*1, a measure considering both *Precision* and *Recall* together, the proposed method shows much better performance.

In the case of F_{12}, which does not include the features developed by experts, the proposed approach shows much better results than the baseline. The *Accuracy*, *Precision*, and *Recall* of the proposed approach are higher by 8.0%, 12.6%, and 4.8% than those of the baseline, respectively. However, in the case of F_{17} for which F_5 is considered together with F_{12}, the performance gap is reduced. The proposed method shows about 2% better performance in the cases of F_{17} and F_5. From this, we conclude that well-developed features from expert knowledge are effective for bot detection.

The performances of the single aspect feature sets are lower than those of the whole aspect feature sets, which shows why we have to consider all aspects of behavior. Among the single aspect features, F_B gives the most accurate results in either methodology. The performance is very close to that of F_{12}. This implies that the battle feature group may represent the most important aspect of player behavior in *Yulgang Online* in terms of clustering and bot detection.

4. Play Style Analysis

In this subsection, we analyze how distinctively the proposed approach discriminates and groups similar play style users and how accurately the proposed approach classifies human and bot players in each play style.

For this purpose, we first identify the play styles of all the players. According to the Bartle Test of Gamer Psychology [26], players in MMORPGs can be classified into four categories, based on their gaming styles, as shown in Table 7.

For the experiment, a group of five human experts play the game more than 100 hours and are given the same raw game playing data of the players. They are asked to identify the players who show strong characteristics toward Killer, Achiever, or Explorer with their own expert knowledge. Player data without any strong behavioral characteristic is simply ignored and later labeled “remainder.” Since there are no features for social behavior in the collected dataset, the experts cannot identify the Socializer player type.

Table 8 shows the player distribution of the three player types and the remainder. The majority of the players, both human players and game bots, make up the remainder. A normal human player may not consistently play in a single play style and thus may not be classified as one of the three player types. On the other hand, game bots continuously repeat the same programmed tasks, so can be more easily classified as one of the styles.

We verify how well the proposed method clusters players with different game play styles. For each feature set, we try four to 14 clusters and select the number of clusters with the best performance. Since the number of clusters for the best bot detection performance may be different depending on the feature sets, the number of clusters in the figures are different. Clustering with F_{17}, the proposed method yields the most accurate results when the number of clusters is four. For each of those four clusters, Fig. 3(a) shows the ratio of Killers, Achievers, and Explorers, but not the remainder, in each cluster. For example, the first cluster (k_1) consists of about 55% Killers, 45% Achievers, and 0% Explorers. The player clustering results by F_{12}, F_5, F_B, F_M, and F_C are also shown in Fig. 3(b) through Fig. 3(f). The number of clusters for the best performance are 12, 5, 6, 4, and 8 for F_{12}, F_5, F_B, F_M, and F_C, respectively.

The clustering result by F_B is the best and the results by F_{17}, F_{12}, and F_5 are relatively good. On the other hand, the results by F_M and F_C are the worst. This outcome confirms that player types are hard to distinguish with only the Move or Collect feature set.

Figure 3(d) by the battle feature set shows an interesting result, in which players are very distinctively separated into different clusters by their play types. This is because the three types of players, Killers, Achievers, and Explorers, have distinct values of battle features. Killers are battle-oriented players, so they have the most extreme values. On the other hand, Explorers are usually quest-oriented players, so they

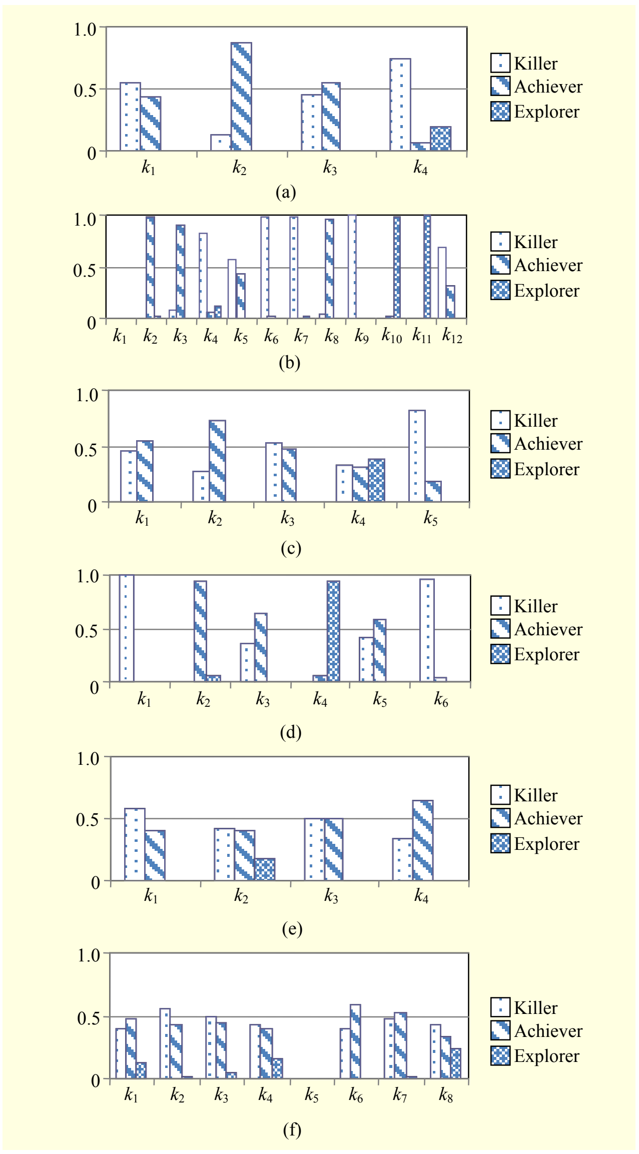

Fig. 3. Player type ratio by (a) F_{17}, (b) F_{12}, (c) F_5, (d) F_B, (e) F_M, and (f) F_C.

have the least extreme values. Even though Achievers engage in battles to gain experiences and collect items, hunting is not the sole purpose of Achievers. The battle-related feature values for Achievers are less extreme than those of Killers but more extreme than those of Explorers.

What is interesting is that play styles are the best separated by F_B, but human and bot players are not well separated compared with the result by F_{17}, F_{12}, or F_5 in Tables 5 and 6. Since players are best clustered by F_B, F_B might give the most accurate results for bot detection. However, the bot detection results are more accurate with F_{17}, F_{12}, or F_5. This also implies that various aspects of player behavior should be considered for not only grouping players by their styles but also analyzing player behavior for bot detection.

Table 9. Accuracy of play styles by baseline.

	Feature	Play style				Dev.
		Killer	Achiever	Explorer	Remainder	
Whole aspect	F_{17}	92.42%	95.57%	95.32%	94.39%	1.59
	F_{12}	86.59%	87.88%	94.89%	82.98%	2.54
	F_5	88.48%	82.70%	96.17%	89.39%	3.63
Single aspect	F_B	86.82%	80.86%	87.23%	77.97%	4.51
	F_M	57.83%	71.46%	94.89%	55.47%	8.63
	F_C	65.64%	66.56%	37.02%	68.72%	1.58

Table 10. Accuracy of play styles by proposed method.

	Feature	Play style				Dev.
		Killer	Achiever	Explorer	Remainder	
Whole aspect	F_{17}	95.35%	96.58%	97.45%	96.30%	0.64
	F_{12}	94.47%	93.26%	94.04%	90.04%	2.29
	F_5	90.21%	85.61%	97.02%	90.48%	2.74
Single aspect	F_B	90.17%	89.12%	86.38%	86.84%	1.70
	F_M	77.25%	74.18%	95.30%	69.89%	3.70
	F_C	69.83%	68.11%	27.66%	68.55%	0.89

Next, we analyze the classification performance of the baseline and the proposed method for each play style. Tables 9 and 10 show the bot detection accuracy for each game play style. For example, for the Killer player group, the baseline has an accuracy of 92.42% with F_{17}, but the proposed method has an accuracy of 95.35% with F_{17}. The proposed approach shows better performance than the baseline in all the cases except the following: Explorer, F_{12}; Explorer, F_B; Explorer, F_C; and the remainder, F_C. Since the Explorer player group is only 1.4% of the total player population, the result may not be meaningful in a statistical point of view, and it does not have much effect on the overall performance.

In addition to superiority in the overall accuracy, the proposed method is more stable. In Tables 9 and 10, the column Dev. shows the standard deviation of accuracy for Killer, Achiever, and the remainder. Since the Explorer player group is 1.4% of the total player population, as mentioned, it is ignored. For example, in the case of F_{17}, the standard deviation of the baseline for Killer, Achiever, and the remainder is 1.59, but that of the proposed method is 0.64. Under any feature sets, the proposed method shows smaller variances than the baseline, which means that the proposed method more stably produces better results. It also shows that the proposed approach is very effective for detection of bots in any play style, regardless of feature sets.

V. Conclusion

In this paper, we proposed a generic bot detection methodology. To detect game bots with different play styles, we examined Battle, Collect, and Move, which are three aspects of player behavior common in MMORPGs and effective for game bot detection in low resolution data. Then, players were grouped by their behavioral similarities. Based on the player groups, the proposed method developed a customized local model of each group for bot detection. Since local models were optimized to detect game bots in each player group, the combination of those models could improve the overall performance.

For comparison, the experiment was performed with the data from a game currently in service. Through the experiment, it was verified that the proposed feature sets were effective and that the proposed local model approach produced more accurate and stable results for all the play styles.

References

[1] P. Hingston, "A Turing Test for Computer Game Bots," *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 3, Sept. 2009, pp. 169-186.

[2] N.W. Lo and S.H. Chen, "A Study of Anti-robot Agent Mechanisms and Process on Online Games," *Proc. ISI*, 2008, pp. 203-205.

[3] P. Golle and N. Ducheneaut, "Preventing Bots from Playing Online Games," *Comput. Entertainment*, vol. 3, no. 3, July 2005, article 3C.

[4] K.T. Chen, H.K. Pao, and H.C. Chang, "Game Bot Identification Based on Manifold Learning," *Proc. ACM NetGames*, 2008, pp. 21-26.

[5] P. Golle and N. Ducheneaut, "Keeping Bots out of Online Games," *Proc. ACE*, 2005, pp. 262-265.

[6] L.V. Ahn et al., "CAPTCHA: Using Hard AI Problems for Security," *Proc. EUROCRYPT*, 2003, pp. 294-311.

[7] A. Cornelissen and F. Grootjen, "A Modern Turing Test: Bot Detection in MMORPGs," *Proc. BNAIC*, 2008, pp. 49-55.

[8] D.Y. Lin, *Anti-robot Agent Mechanisms on Online Games — A Game Level Approach*, master's thesis, National Taiwan University of Science and Technology, 2007.

[9] E. Castronova et al., "As Real as Real? Macroeconomic Behavior in a Large-Scale Virtual World," *New Media Soc.*, vol. 11, no. 5, 2009, pp. 685-707.

[10] C. Platzer, "Sequence-Based Bot Detection in Massive Multiplayer Online Games," *Proc. ICICS*, 2011, pp. 1-5.

[11] A.R. Kang et al., "Online Game Bot Detection Based on Party-Play Log Analysis," *Comput. Math. Appl.*, vol. 65, no. 9, 2013, pp. 1384-1395.

[12] R. Thawonmas, Y. Kashifuji, and K.T. Chen, "Detection of MMORPG Bots Based on Behavior Analysis," *Proc. ICACET*, 2008, pp. 91-94.

[13] S. Mitterhofer et al., "Server-Side Bot Detection in Massively Multiplayer Online Games," *IEEE Trans. Sec. Privacy*, vol. 7, no. 3, May-June 2009, pp. 29-36.

[14] G. Mori and J. Malik, "Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA," *Proc. CVPR*, 2003, pp. 134-141.

[15] J. Yan and A.S.E Ahmad, "Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms," *Proc. ACSAC*, 2007, pp. 279-291.

[16] M.L. Mauldin, "CHATTERBOTs, TINYMUDs, and the Turing Test: Entering the Loebner Prize Competition," *Proc. NCAI*, 1994, pp. 16-21.

[17] K.T. Chen et al., "Identifying MMORPG Bots: A Traffic Analysis Approach," *Proc. ACE*, 2006.

[18] M.V. Kesteren, J. Langevoort, and F. Grootjen, "A Step in the Right Direction: Botdetection in MMORPGs Using Movement Analysis," *Proc. BNAIC*, 2009.

[19] S. Hilaire, H.C. Kim, and C.K. Kim, "How to Deal with Bot Scum in MMORPGs?" *Proc. CQR*, 2010, pp. 1-6.

[20] M.A. Ahmad et al., "Mining for Gold Farmers: Automatic Detection of Deviant Players in MMOGs," *Proc. CSE*, 2009, pp. 340-345.

[21] K.T. Chen et al., "Game Bot Detection Based on Avatar Trajectory," *Proc. IFIP ICEC*, 2008, pp. 94-105.

[22] M. Varvello and G.M. Voelker, "Second Life: A Social Network of Humans and Bots," *Proc. NOSSDAV*, 2010, pp. 9-14.

[23] C. Thurau and C. Bauckhage, "Analyzing the Evolution of Social Groups in World of Warcraft®," *Proc. CIG*, 2010, pp. 170-177.

[24] A. Mislove et al., "Measurement and Analysis of Online Social Networks," *Proc. SIGCOMM*, 2007, pp. 29-42.

[25] C.J. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining Knowl. Discovery*, vol. 2, no. 2, June 1998, pp. 121-167.

[26] R. Bartle, *Designing Virtual Worlds*, Indianapolis, IN: New Riders Publishing, 2004.

Yeounoh Chung received his BS in electrical and computer engineering and his ME in computer science from Cornell University, Ithaca, NY, USA, in 2008 and 2009, respectively. He is currently a researcher at Sungkyunkwan University, Suwon, Rep. of Korea. His current research interests include statistical machine learning and data mining.

Chang-yong Park received his BS in computer engineering from Dongguk University, Gyeongju, Rep. of Korea, in 2010. He is currently pursuing his MS in computer engineering at Sungkyunkwan University, Suwon, Rep. of Korea. His research interests include software engineering, context-aware recommender systems, and latent variable modeling.

Noo-ri Kim received his BS in computer engineering from Sungkyunkwan University, Suwon, Rep. of Korea, in 2013. He is currently pursuing his MS in computer engineering at Sungkyunkwan University. His research interests include software engineering and latent variable modeling.

Hana Cho received her BS in computer engineering from Sungkyunkwan University, Suwon, Rep. of Korea, in 2013. She is currently pursuing her MS in computer engineering at Sungkyunkwan University. Her research interests include data mining and artificial intelligence.

Taebok Yoon received his BS in computer science from Kongju National University, Gongju, Rep. of Korea, in 2001. He received his MS and PhD in computer engineering from Sungkyunkwan University in 2005 and 2010, respectively. He is now an assistant professor at Seoil University, Seoul, Rep. of Korea. His research interests include user modeling, intelligent systems, and game artificial intelligence.

Hunjoo Lee received his BS, MS, and PhD in computer science and engineering from Chung-Ang University, Seoul, Rep. of Korea, in 1991, 1993, and 1998, respectively. In 1998, he joined ETRI, Daejeon, Rep. of Korea. He was a postdoctoral researcher at Iowa State University, Ames, IA, USA, from 2001 to 2002. His current research interests include game artificial intelligence, serious games, and smart content.

Jee-Hyong Lee received his BS, MS, and PhD in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Rep. of Korea, in 1993, 1995, and 1999, respectively. From 2000 to 2002, he was an international fellow at SRI International, USA. He joined Sungkyunkwan University, Suwon, Rep. of Korea, as a faculty member in 2002. His research interests include fuzzy theory and application, intelligent systems, and machine learning.