STAIRS 2014
U. Endriss and J. Leite (Eds.)
© 2014 The Authors and IOS Press.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-421-3-230

Predicting Players Behavior in Games with Microtransactions

Ondřej Pluskal [a] Jan Šedivý [a]
[a] *Czech Technical University in Prague, Czech Republic*

Abstract. This paper focuses on predicting player behaviour in two-player games with microtransactions. Typically the games are for free and companies generate their revenue by selling in-game goods. We show creation of a users behaviour model, which are then used in a recommendation system increasing in-game goods purchases. We focus on learning techniques in a novel way, predicting the time of purchases rather than the most likely product to be purchased. The player model is based on in-game signals, such as players success, curiosity, social interactions etc. We had access to a *Pool Live Tour* game dataset made by Geewa. We report promising results in predicting the purchase events.

Keywords. machine learning, feature extraction, online games, data mining

1. Introduction

In this paper we specifically focus on how to improve the monetization of two-player online games. The game is typically free but to gain a special game advantage players may purchase in-game goods improving their skills. They purchase the goods in small payments, microtransactions, for real-currency. To increase the revenue users are bombarded with ads [5]. The problem is that very frequent advertisements are do not typically lead to increased revenue because of advertisement fatigue and advertisement wear out.

Both phenomenons are recognized in marketing models [9]. Advertisement fatigue is the event of a customer no longer liking or buying goods from the advertisement, because the advertisement is bothering them too much. Advertisement wear out means that the customer will ignore the advertisement and it would have no effect.

These two negative effects can be reduced by advertising the in-game goods only when the player is likely to make the purchase. When the ad's timing is correct the likelihood of converting through the advertisement is increased. This improves the revenue. Building such a system by an expert is impossible, simply because we are analysing a large-scale dataset of thousands of players.

We tested our novel idea on a *Pool Live Tour* (PLT) game made by Geewa. PLT is a virtual pool game, that can be played in a web browser or on a tablet. This game is played by 2.5 million daily active users[1] all over the world. The in-game good are better cues giving a slight advantage. Certainly, buying a better cue does not guarantee a win it only increases player's chances. To design the model Geewa provided us with

[1]Collected from http://corporate.geewa.com/game/pool-live-tour/ on 2nd Feb 2014

two datasets, dataset D_1 with a sample of monthly users' activity, totalling 272k unique user ids, and dataset D_2 with monthly activity of a subset of users, that registered that month and bought in-game item, totalling 11.5k unique user ids. The datasets contain client actions collected from the clients as well as events generated by the server. The datasets in their raw form contain over 30 GB.

We show creation of a model from the player logs that predicts the players' readiness to buy a cue after finishing a match. In the next section we review the behaviour prediction in general. We will also discuss impact of some of the recommendation systems. In Section 3 we explain our approach to the behaviour prediction as a machine learning task and develop and evaluate the model. In Section 4 we show the properties of the datasets and how the feature extraction was done. In Section 5 we describe the testing and show the results of our experiments. We conclude our observations and set our goals for future work in Section 6.

2. Related Work

In the literature there are a few studies concerning player modelling using machine learning and data mining methods. We have identified a few trends in the literature, from matchmaking, user segmentation, to cheater detection. Up to our knowledge, nobody published any work about building recommendation systems in games. This section therefore covers not only player modelling, but also recommendation systems.

In cheater detection studies there are several studies using machine learning and data mining methods. The studies define the problem as a classification problem. They use derived rules [7], SVM [13] and Hidden Markov Models [16] with a set threshold. The features were consisting of playtime lengths[7] or action frequencies [13]. Bot detection tasks are studied in several game settings. These are mainly MMOGs[2] [7,13] and FPS[3] games [16].

More descriptive models were made in the case of segmentation [3,4,14], a problem of finding distinct groups of users. Segmentation is viewed in all of these articles as unsupervised machine learning problem. Two papers consider player segmentation task [3,4], the third segments organised groups of players called guilds [14]. Player segmentation is done using cumulative features including playtimes, game specific estimates of success (e.g. number of trials per level, number of kills etc.). The approaches used for this task vary from NMF[14], k-means, Simplex Volume Maximization[3], to SOM[4]. In research the main challenge is the number of users and therefore the problem of large-scale datasets. All the interesting information are extracted from game logs, that the producers store in databases acquired by telemetry. The games used in player segmentation are both single player games [4] as well as multiplayer games [3,14].

There are more papers[10,15] using supervised machine learning. Another paper tries to predict the players' maximal progress in a single player game [10]. This is done using machine learning methods, by monitoring gameplay features in the early levels of the game and solving this problems both as classification problem as well as regression problem. Both problem abstractions show promising results. Another paper [15] studies

[2]Massive Multiplayer Online Games
[3]First Person Shooter

the prediction of strategy that a competitive RTS[4] player would play from his actions done in game at certain time point. This study aims to creating an AI that by predicting the strategy probabilities would infer correct counter strategy.

Since up to our knowledge in the research of player modelling there is no relevant literature of how to present players with in-game goods we have tried to find resources in the setting of web recommendation systems aimed at recommending relevant goods, entertainment or web pages to specific users based on their preferences or browsing history.

In recommendation systems there are many ways from which data we should predict what a user might like. This can be done via contextual information, where in advertisement we try to find ads with similar context to the site. Also user based recommendations, where we look at similar users and recommend goods to them that others purchases (Collaborative filtering). And the last approach is that we use sequential data and try to find sequential patterns that might help us what the user might want to see next. We try to find out similarities with web recommendation, which is in active research for more than ten years[11]. Also hybrid approaches [8] using combinations of different approaches can be seen mainly in the Netflix Prize competition, where researchers improved the accuracy of the Netflix recommendation system by 10%.

Other approaches use sequential patterns [17,12]. The idea comes from web-page recommendation, where a users browsing in some domain might share the same browsing sequential patterns as other users. Finding the most common patterns and creating a patricia trees from them is the core idea. Then assuming the Markov property (dependence of next state only on previous state, here states are web pages) the recommendation system can show the most probable web pages by traversing the patricia tree and suggesting the most probable pages.

To the best of our knowledge there exists no study that would directly solve the issue of dynamic advertisement space generation presented in this paper. This approach can bring the notion of smart recommendation to games, where we can address the player directly in the time, he is most likely to buy a new virtual item.

3. TASK DEFINITION

The main motivation of this paper is to increase the revenue generated by games using microtransactions by reducing the amount of advertisement sent to the user. In the ideal case we would like to suggest to the user to purchase in-game items only in the times he would really buy the in-game item.

We will be explaining our approach on the PLT game. In this game there are two currencies. The first one is coins, that can be acquired by playing, daily rewards and earning trophies. The second one is gold coins, that can be only acquired by purchase using real currency. The task at hand is predicting the buy of a cue for gold coins. We divided the task into two stages, first we try to predict that the player would buy a cue of any kind, then we decided to only predict the gold cue buys. Both stages use only users of which we have information from their registration. For the second stage we use a segment of paying users as a subset of all users whose behaviour we would like to model.

[4]Real Time Strategy

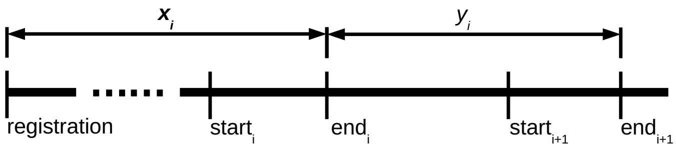

Figure 1. Visualization of $\mathbf{x}_i$ and y_i for player u.

We have decided to model the problem of timely in-game item recommendation as a supervised machine learning task, particularly a binary classification task. The example is a tuple $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ corresponds to a feature vector containing information about the activity of player u from registration to the end of game i. Fig 1 depicts the $\mathbf{x}_i$, y_i relationship on an example of a single user. The y_i is whether the player u bought a cue in between the start end of game i and the end of game $i+1$. Because we have only two targets, we define $y_i = -1$ if the player didn't buy between games i and $i+1$, $y_i = 1$ if the player bought between games i and $i+1$.

Since we described the task as a binary classification tasks, we can use several measures to score classifiers performance. We could use accuracy, precision and recall or area under a ROC curve (AUC) [6]. We decided to use AUC because it generally maximizes the True Positive Rate (TPR) for every False Positive Rate (FPR). This is a good measure for this problem, because upon creating the recommendation system the operator can choose the FPR and then set a threshold on the scoring decision function.

4. DATA COLLECTION

Two datasets of raw data in form of logs were provided by Geewa from their game PLT. The data come from their internal reporting system, that is used by their analytics in order to better understand how the players are playing the game.

These are live data from real players in their homes captured by telemetry and stored on Geewas side. The datasets contain logs from the client residing in player's computers as well as actions generated by the server. The player base is from users from 187 countries all over the world. We were kindly provided with 2 datasets D_1 and D_2 from Geewa. D_1 is drawn as a small sample of all users and all of their actions generated by client and server in one month (19th Mar 2013 - 18th Apr 2013). D_2 is a set of all users that registered and bought some cue in one month (1st Nov 2013 - 30th Nov 2013). The size of raw data is 24 GB for D_1 and 6.6 GB for D_2.

4.1. Data Preprocessing

The log data contain information about user logins, their in-game actions, from starting a game to shooting with a cue. The dataset was provided in a single csv file. We had to split the single file into a file for each player and those needed to be sorted according to the time when they were generated. Both datasets are exported from relational databases in form of one large table with all events in a structured form.

The dataset D_1 consists of a subset of all users. This means that out of 272k users for example we have 181k users who played one game and 78k users who played 10 games. Since we need full user history, we need the users who have newly registered, which out of the total 272k users is only 57k users who registered. Only 6728 players have played at least 10 games, newly registered and bought some cue. These players were used for the first task of prediction of cue buys.

The dataset D_2 consists of a subset of players who bought a gold cue. This process unfortunately yielded only 6,838 players who registered and played at least 10 games.

4.2. Feature Extraction

We will now describe how we created the feature vector $\mathbf{x}_i$. We created a set of features extracted from the logs containing information about the players progress, success and curiosity. Also there is a difference in over which period the features are extracted. They can be extracted from the registration until the game end, but they can also be extracted from several games before the game end or even in time periods, e.g. in the last 5 hours.

We define a set of all basic features F. These are the basic features extracted from the logs, that are in F:

- **Number of matches:** Number of matches m_a the player played in both main game modes, matched and friend games.
- **Distribution of match types:** Number of each game type the player played. There are 2 different match types in PLT. These are matched games m_m (the player is matched against an opponent according to his skill), friend games m_f(player challenges his friend through some social platform).
- **Distribution of levels played:** A player can play matched games in different game levels in the play mode. There are levels present at the moment. The extracted feature $m_i \; \forall i \in \{1,2,...,14\}$ is the number of games played at each level.
- **Performance in matches:** The number of wins w in different game types. From the game point of view each of the two competitive game types (matched games w_m and friend games w_f) the number of wins for both match types w_a. Also the number of wins per level in matched games is measured w_i. In matched games the player bets a fixed amount of coins and wins his and his opponent's bet, or loses everything. The stakes are the higher the higher is the level.
- **Match properties:** Number of shots m_{ns}, time spent in game t_m.
- **Player curiosity:** Number of shown opponents cards c_{pc}, cue galleries c_{cg}, owners card c_{oc}, shop c_s.
- **Number of trophies:** Number of trophies a a player got through achievements (e. g. winning several games in row, sinking several balls in a row etc.). The player also gets a small reward of additional coins for each trophy.
- **Bonus coins:** A player can get coins for free in two ways, by receiving a daily-bonus package c_{db} or by depleting all coins and receiving a free-bonus package c_{fb}.
- **Amount of coins:** Current coin balance c_b. The coins are used to bet and play games. The bet value is set at each level and if the player wins, he gets back both bets of both players. If the player loses he gains nothing. The player can spend coins for various in-game items, for example cues. Also the player can buy coins for real currency.
- **Amount of winnings:** The cumulative amount of gold a player has won is used to unlock levels. For each level there is a set threshold for winnings, if the player

exceeds the threshold he has the option to play at that level. The feature is denoted as c_w.
- **Rank-ups:** When a player is able to play at a higher level (he has just unlocked the level or he has enough coins to play at a higher level) he is shown a dialog, that he is able to play at higher level. The number of these shown dialogs is encoded in feature c_r.
- **Time from registration:** How much time m_t passed between the end of game and registration.
- **Inventory:** Number of already bought cues b_c and the subset of gold cues b_{gc}.

As we said the feature extraction can be done in three different ways. We will be explaining them using a generic feature f. One is by extracting the cumulative features from registration up until game i, $f(i)$. The second one is by extracting the features over the last k games, $f^{(k)}(i)$. We call this extraction method match windowing. The third is extracting features over a certain time period t, $\hat{f}^{(t)}$. We call this time windowing.

The feature vector is described in equation 1. The t time in time window features $\hat{h}_i^t$ is in hours.

$$\begin{aligned}
x = \; & (\, f_1,\dots,f_n, f_1^{(1)},\dots,f_m^{(1)}, f_1^{(10)},\dots,f_m^{(10)}, \\
& \hat{h}_1^{(1)},\dots,\hat{h}_l^{(1)}, \hat{h}_1^{(24)},\dots,\hat{h}_1^{(24)}, \hat{h}_1^{(168)},\dots,\hat{h}_l^{(168)}) \\
f_i \in \; & F \cup R \\
h_i \in \; & \{\, m_a, m_m, m_f, m_1,\dots,m_{14}, \\
& w_a, w_m, w_f, w_1,\dots,w_{14}\} \cup R
\end{aligned} \tag{1}$$

We are using a smaller feature vector for the games played before the first 10 games, since for computing the features $g_i^{(10)}$ we need to have at least 10 games played by the user. We divide each of the datasets D to two parts, $D^{(1-9)}$ and $D^{(10)}$, where feature vectors from $D^{(1-9)}$ do not have the $f_i^{(10)}$ features, but the rest is the same as described in equation 1.

5. EXPERIMENTS

The methodology used for evaluation is based on measuring the AUC. In order to create a valid training protocol, we divided the dataset into two parts, training set and testing set, where training set is 20% and testing set is 80%. If parameter tuning is needed in order to find the best model we use 5-fold cross-validation.

5.1. *User based and game based dataset divisions*

The most important issue when constructing a testing protocol is the proper division into training, validation and testing sets, so that the examples are i.i.d.(independent and identically distributed). In this task we can split either based on user, or based on games. In the previous section we described how to extract the tuples $D = (\mathbf{x}_i^u, y_i^u), \forall u, \forall i$. The split based on games would be done, by considering every tuple $(\mathbf{x}_i^u, y_i^u)$ independent on u. This could lead to having a feature vector from particular user u in both training and testing set, i.e. game 11 in training and game 12 in testing set. But this contradicts with

Table 1. This table shows the difference between validation and testing error with different splits. It ilustrates overfitting with different techniques.

	Validation AUC	Testing AUC	Change
Game split	74.76%	73.06%	-1.7
User split	75.07%	75.99%	+0.92

Table 2. This table consists of the properties of each of the different datasets.

	D_1^{1-9}	D_1^{10}	D_2^{1-9}	D_2^{10}
No. buys	6,838	11,601	445	3750
No. games	83,160	383,703	35,271	445,554
No. players	9,240	6,161	3,919	3,511

the independency of the two sets. Also we suggest that the performance on new users would be lower for the game split.

We made an experiment to support our claims and the results are shown in table 1 using a Random Forest on $D_1^{(10)}$. The dataset was divided into a training set and testing set using a user split. The validation error is the error using 5 fold cross-validation using the respective splitting on training set. The testing error is the performance of the classifier trained on the whole training set evaluated on the testing set. We can see that our assumption were confirmed empirically.

5.2. General cue and gold cue predictions

Since we are not capable of computing the window features for the early matches, we decided to divide the dataset into two parts. One are games 1-9 and the second part are the games 10 up to what the player accomplished to play in given month. The dataset will be labelled $D^{(1-9)}$ and $D^{(10)}$.For dataset D_1 we will perform the experiment for all cues, since there are only 299 gold buys. For dataset D_2 we will be making experiments on gold cues. The feature vectors will be the same for both settings.

From the characteristics of the dataset shown in table 2, we can see that the datasets have a property of being unbalanced. This means that we have to choose the classifiers accordingly. We have chosen scikit[5] library in python for the training and evaluation of the classifiers.

As the classifier to test the performance we have chosen Random Forest [1], linear SVM [2] and a Decision Tree. They were chosen, because they have the option to handle unbalanced classification tasks, due to the ability to assign class weights. The class weights used were inversely proportional to class frequencies. The Random Forest is a well performing classifier in various different tasks and linear SVM represents a simple but well grounded model. For the classification task we used a small decision tree in order to create a simplified view of the problem. The Decision Tree has maximum depth set to 3 in order to give the reader an idea what are the most discriminative features and how do they work.

Z-normalization was performed on the respective training sets when training the SVM, because the scales in each feature are different. The scales do not affect the tree

[5]Version 0.14 http://scikit-learn.org/stable/

Table 3. Results of the experiments for each classifier and dataset.

	Random Forest	SVM	Decision Tree
$D_1^{(1-9)}$	75.37%	71.22%	71.53%
$D_1^{(10)}$	75.99%	73.25%	68%
$D_2^{(1-9)}$	79.69%	78.65%	76.96%
$D_2^{(10)}$	87.22%	76.72%	78.27%

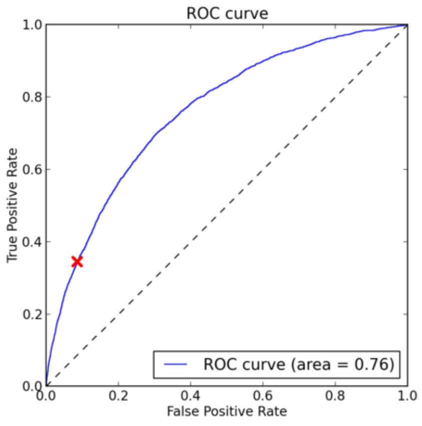

Figure 2. ROC curve of $D_1^{(10)}$

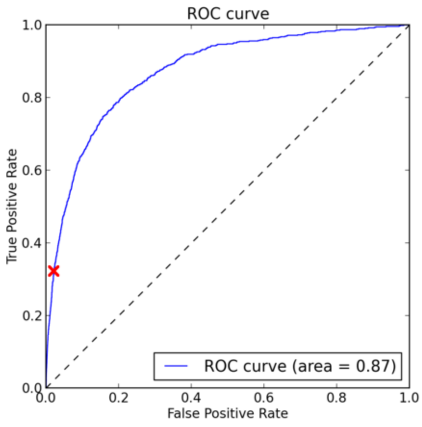

Figure 3. ROC curve of $D_2^{(10)}$

based classifiers. SVM's regularization constant C was tuned on the validation sets. The parameters tuned on the Random Forest were maximum number of feature, minimum number of features and number of estimators.

In table 3 we can see the performance of each classifier on each of the two datasets. We can see that the performance on the first dataset is significantly lower than the performance on the second dataset. We suppose the reason for higher performance of Random Forest over SVM is the fact, that the dataset is clearly not linearly separable. The Random Forest can infer much more complex models than linear ones [1]. Also we can see that the models' performance using only a small decision tree is significantly lower that each of the Random Forests.

In Figures 2 and 3 we can see the ROC curves for all the datasets computed over the testing set. In each figure the cross is the operating point of the classifier.

Both ROC curves show, that the classifiers default operating point was trained on a very low FPR. This corresponds to not suggesting the cues to the player many times. An operator can adjust the threshold to move the TPR as well as FPR higher, in order to raise the advertisement ratio.

The relative feature importance can be computed from each of the trained random forests[1]. We are plotting only the first 20 highest scoring features in order to see, what features are most important and to check whether all the types of designed features are used by the random forest. In figures 4 and 5 we can see, that all types of different features are present. These include to cumulative, time window and game window features.

For both cases $D_1^{(10)}$ and $D_2^{(10)}$ we can see similar features scoring high. One exception is feature importance of the number of previously purchased cues b_{gc} is 0.19 in

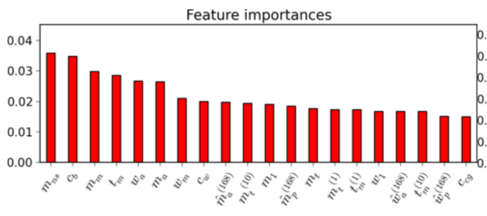

Figure 4. Feature importances of $D_1^{(10)}$

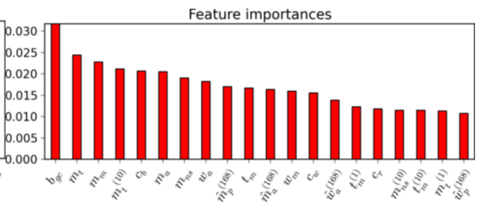

Figure 5. Feature importances of $D_2^{(10)}$

figure 5. This is due to the fact, that many players buy only one cue and the acquisition of a second cue is for the most cases unlikely. Another positive result is the appearance of many different types of features, including cumulative, time window and game window features.

Also we can see that among the highest scoring features there is no feature considering friend matches. The reason is that friend matches are less common than matched matches. In the general cue case we can see that an important feature is the number of clicks on opponent cue gallery. From opponent's cue gallery he can see he is playing against a player with better cue. Using these feature importance plots also might indicate to the developers key components of the game that lead to possible design changes.

6. CONCLUSION AND FUTURE WORK

We presented a system predicting player purchase applicable on games with microtransactions. Unlike a typical recommendation system we are not solving the problem of what we should recommend to the player, but when is the right time to recommend purchase of in-game goods. This approach allows dynamic advertisement placements.

The task is defined as a classification problem deciding whether to display or not an advertisement at the end of a game. This decision is based on the prediction whether a player would buy an in-game item after a match. The features used are of three types, cumulative, game windows and time windows.

We created models predicting two actions general in-game items purchases and in-game items purchases for hard currency leading to real revenue. Our experiments have shown that the Random Forest algorithm was performing the best for both cases. For the general case we have achieved 76% AUC and for the hard currency 87% AUC. We have used the decision trees to study features differentiating power. We have found that the most discriminative features are different for each case. Information about feature importances can be useful to the game designers in order to improve the game and where to focus their design goals.

This system is yet to be tested in practice, because the only way how to measure the real impact on revenue is by A/B testing. We optimistically hope that the correct advertisement placement timing will make the players buy the in-game items for hard currency more frequently or earlier than today. This will bring higher revenue to the company.

We plan to use these models in other tasks such as cheater, bot detection and churn prediction in the future. The developed models can help to solve also these big problems.

7. ACKNOWLEDGEMENTS

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS14/072/OHK3/1T/13.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[2] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, June 1998.

[3] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau. Guns, swords and data: Clustering of player behavior in computer games in the wild. In *Computational Intelligence and Games (CIG), 2012 IEEE Conference on*, pages 163–170, 2012.

[4] Anders Drachen, Alessandro Canossa, and Georgios N. Yannakakis. Player modeling using self-organization in tomb raider: underworld. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, CIG'09, pages 1–8, Piscataway, NJ, USA, 2009. IEEE Press.

[5] Canossa Alessandro. El-Nasr Magy Seif, Drachen Anders. *Game Analytics: Maximizing the Value of Player Data*. Springer, 2012.

[6] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[7] Ah Reum Kang, Jiyoung Woo, Juyong Park, and Huy Kang Kim. Online game bot detection based on party-play log analysis. *Computers & Mathematics with Applications*, 65(9):1384 – 1395, 2013. Advanced Information Security.

[8] Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.

[9] Moorthy S. Lillien G., Kotler P. *Marketing Models*. Prentice Hall, 1992.

[10] Tobias Mahlmann, Anders Drachen, Julian Togelius, Alessandro Canossa, and Georgios N. Yannakakis. Predicting player behavior in tomb raider: Underworld. In *CIG*, pages 178–185. IEEE, 2010.

[11] R. Suguna and D. Sharmila. An efficient web recommendation system using collaborative filtering and pattern discovery algorithms. *International Journal of Computer Applications*, 70(3):37–44, May 2013. Published by Foundation of Computer Science, New York, USA.

[12] Usha Rani M. Suneetha K. Web page recommendation approach using weighted sequential patterns and markov model. *Global Journal of Computer Science and Technology*, 2012.

[13] Ruck Thawonmas, Yoshitaka Kashifuji, and Kuan-Ta Chen. Detection of mmorpg bots based on behavior analysis. In *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, ACE '08, pages 91–94, New York, NY, USA, 2008. ACM.

[14] C. Thurau and C. Bauckhage. Analyzing the evolution of social groups in world of warcraft. In *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, pages 170–177, 2010.

[15] Ben G. Weber and Michael Mateas. A data mining approach to strategy prediction. In *Proceedings of the 5th international conference on Computational Intelligence and Games*, CIG'09, pages 140–147, Piscataway, NJ, USA, 2009. IEEE Press.

[16] S.F. Yeung, J.C.-S. Lui, Jiangchuan Liu, and J. Yan. Detecting cheaters for multiplayer games: theory, design and implementation[1]. In *Consumer Communications and Networking Conference, 2006. CCNC 2006. 3rd IEEE*, volume 2, pages 1178–1182, 2006.

[17] Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. An intelligent recommender system using sequential web access patterns. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 393–398 vol.1, 2004.