Learning to Recommend Game Contents for Real-Time Strategy Gamers

Hyun-Tae Kim and Kyung-Joong Kim*
Dept. of Computer Science and Engineering
Sejong University
Seoul, South Korea
kimhyun0416@sju.ac.kr, kimkj@sejong.ac.kr

***Abstract*— It is not surprising that there are currently many game videos, increasingly prevalent on the web, to watch professional player's matches. Therefore, we need a kind of recommendation system to select the most preferable contents. Although there have been many personalization techniques proposed, there are few works on the recommendation of personal game contents. In this paper, we attempt to propose to use machine learning based on game contents with user's explicit preference (like or dislike). Especially, we incorporate game domain knowledge on the design of features for learning. As a test bed, we select a famous real-time strategy game, StarCraft, because there are many game replays played by professional players. To generate training samples, each participant is invited to review replays and express his/her preference. Using the data, machine learning algorithms build a set of models to predict preference on new replays. For five participants, we labeled two hundred StarCraft replays. The exploratory study on the selection of the features and classification algorithms show the importance of the careful selection of them. The experimental results show that the proposed recommendation system can predict the users' preference with an accuracy of 79% when we select appropriate models and features.**

Keywords*— *StarCraft; Data Mining; Recommendation; Game Contents;

I. INTRODUCTION

Exponential growth of content has caused a significant problem in terms of the access of information for novice users. For example, there are thousands of books, music, and movies available on online markets (for example, Amazon) but it is not always easy to find the best items suitable for users. It is essential to understand users and provide an efficient way to find products from the millions of goods. In this line of research, there has been a lot of research to make recommendations: Amazon's successful collaborative filtering approach [1] and Netflix[1] awards to find better algorithms.

Recently, games are not just enjoyable computer programs but also a kind of culture with a lot of derived contents. For example, users buy weapons, items, cloths, and virtual flowers to increase enjoyment and save the time needed to get them. Also, there are large markets to exchange game items, as well as broadcasting channels on the popular game players and social communities to share their interest on games. Game players generate a lot of related content and they are uploaded on social network services. It makes a huge amount of content related to games, and therefore novice gamers may suffer from searching for suitable contents [2].

In this paper, we focus on the real-time strategy games which have gained popularity in the public. There have been a lot of broadcasting channels on the games and their replays have been shared through internet portals. It is easy to download thousands of game replays played by professional gamers for StarCraft (Figure 1). In this work, we attempt to build a recommendation system to select interesting replays from a pool of them. Although it is possible to recommend replays simply based on the player's name or levels, there could be many complex factors (specific gaming situation, special interest on units, high-level strategy and so on) on the selection of enjoyable games [3].

Figure 1. A screenshot of the StarCraft : Brood War

The goal of this research is to extract features from StarCraft game replays and try to learn a model to explain the user's mind. Like other recommendation systems, this system initially requires explicit feedback on game replays enjoyed by users. When a user watches the games, he/she explicitly marks

[1] http://www.netflix.com

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (2013R1A2A2A01016589, 2010-0018950).

*Corresponding author

978-1-4799-3547-5/14/1.00 ©2014 IEEE

the interesting points during the games. Based on the number of markings, it is possible to identify the games in terms of either "like" or "dislike." Based on our earlier works [4],[5] on the "strategy prediction" of games, we expanded some new features to predict the user's interest on replays.

To the best of our knowledge, there are few works on the personal recommendation of game contents using machine learning. There has been a system to evolve a flower shape based on user feedback with evolutionary algorithms. In the work, they created a flower for social games using evolutionary neural networks [6]. Recently, Togelius *et al*. [7] evolved a set of game maps for StarCraft via multi-objective evolutionary algorithms. They defined objective measures to rank goodness of randomly created game maps. Although they showed the usefulness of automatic content generation, their contents were not tailored to the each user's preference. Also, the interest of audiences on games of a live video streaming platform (called TWITCH) is analyzed for a first characterization of new web community [8]. But, they have only used data for analyzing the popularity of the game. In this paper, we attempt to suggest new preferable game matches (not new game contents generated by automatic contents generation) to users using a data mining approach.

The contribution of this paper is to introduce the machine learning approach to the recommendation of game contents. It is not designed to create interesting new contents based on objective measures (closeness to minerals and resources, symmetry, and so on). The system is purely based on user's explicit feedback on the likeness of game replays. The most difficult part is to extract features useful to predict a user's preference and we attempt to test a lot of different combinations of features. The experimental results on two users show that the learning algorithms can predict user's interest with an accuracy of 80%.

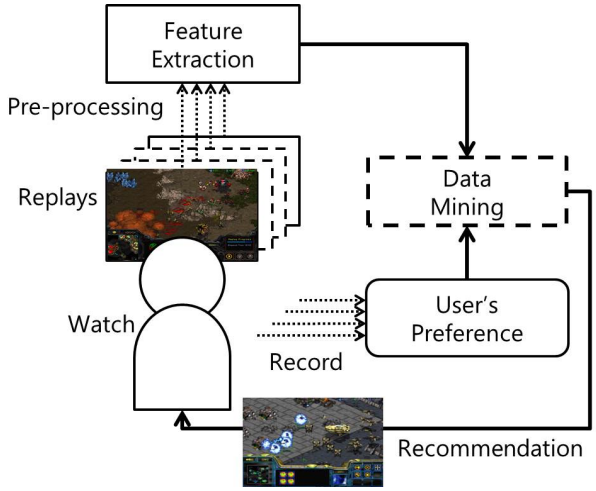

Figure 2. An overview of the proposed game contents recommendation system

Our proposal in this paper is to exploit a large number of replay video game content and to implement the personalized game-contents recommendation system (PGRS) with game content analysis (Figure 2). Actually, in terms of the contents type and appropriate recommendation methods, the PGRS seems to be a TV and video recommendation system. However, game contents require more domain knowledge about the game than TV and video contents. For example, there are a lot of game units and building a specific character of a game unlike in the genre of videos. As a result, the introduction of the automated machine learning approach is promising.

II. Background and Related Works

A. StarCraft Strategy

In RTS (Real-Time Strategy) games, building an effective strategy to beat opponents is very important and expert players usually spend much time in order to optimize it. In fact, the design of strategy is not simple but includes a lot of different factors to be considered. Furthermore, players should identify the opponent's strategy during the games to select the best one. The strategy in the early stage of the game is characterized with a "build-order", the order to create units and buildings.

To detect an opponent's game strategy, the player has to scout an opponent's territory while playing the game, because you can see only near the friendly units (fog-of-war). The "Fog" in games increases the uncertainty of the game playing. Thus, players have to decide their strategy and behavior under uncertainty [3],[9]. Because of the uncertainty, it is possible to design an interesting strategy for players and the audiences on professional player's games experience much enjoyment. To minimize the effect of the "fog-of-war" and the uncertainty, AI bots are developed for generating group position [10] or effective scouting to probe an opponent's strategy [5]. In fact, an AI bot with an effective scouting system has a higher percentage of wins than other AI bots [9].

However, in contrast with an AI bot, viewers who watch the game matches are interested in various strategies from uncertainty rather than in the uncertainty. Although there are many factors in games other than just strategy, the strategy represents the player's intention on RTS games. Weber and Mateas [11] used labeled replay files for the classification problem to predict an opponent's strategy. They labeled the game replay with a strategy using rules based on analysis of expert play, and perform a data mining task with labeled replays and the attributes represented as game event. Therefore, we also used the attributes (the sequence of game events) to represent a player's strategy in the match and detect the viewer's intention from the game match.

B. Replay-based StarCraft Strategy Prediction

Classical board games (like Chess, Go, and Othello) open all information on the boards to the players. It is possible to cope with different situations and change their strategy considering the opponent's strategy. But, it is very difficult to judge the current situation and control their units in an RTS game with the fog of war, because they play games under uncertainty. With the fog-of-war, it is impossible to know the position of the building and units all the time. Usually, the player is able to know a part of an opponent's colony or not at all. To predict the opponent's strategy based on the given

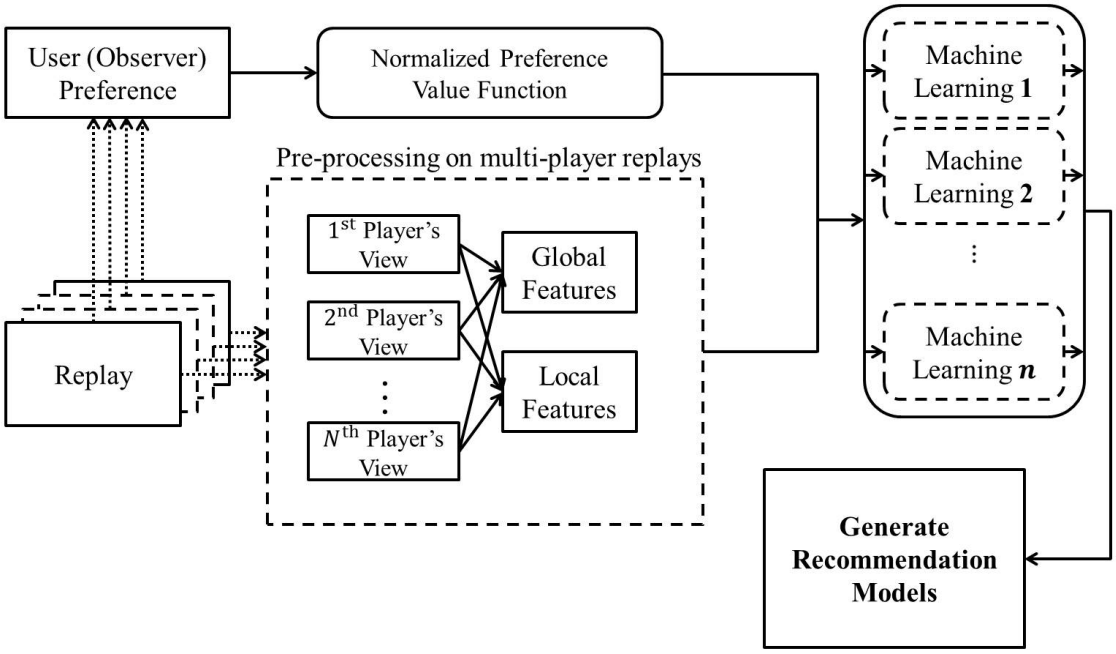

Figure 3. Schematic diagram of the proposed method

information, the player has to remember the previous situation or uncover parts of the current status of the opponent's colony through scouting.

Recently, there are some works on the replay-based prediction of opponent's strategy for AI bots. For example, Cho *et al*. used machine learning to predict the player's strategy with the "fog-of-war." They reported that the prediction with "fog-of-war" is more difficult than one without "fog-of-war" [4]. Gabriel *et al*. proposed a Bayesian model for strategy prediction in an RTS game on real-time. This model predicts a possible opponent's strategy [12]; also they presented an unsupervised learning Bayesian model for tech-tree prediction from noise observation. The proposed probabilistic model computes the distribution over an opponent's build-order in a RTS game [13].

III. PROPOSED METHOD

This section presents the proposed method to extract new features and build recommendation models from replay video game contents for the purpose of this study. Our implemented method is based on the user's initial explicit and subjective feedback of game replays. We found that it is not easy to express user's feedback into one of "like" or "dislike." Instead, we ask users to mark the interesting points during the games and use the statistics to measure the "likeness" of the replays.

In game strategy prediction [4], it is important to define features to express the "build-order" of each player. However, in the personal game contents recommendation, it is challenging to design complex feature sets to reflect a user's impression. In this work, we attempt to design combinatorial feature sets from a pool of high-level and low-level features.

Figure 3 shows the overview of the proposed method for personalized game contents recommendation. From the game community portals, a user can access thousands of game replays and it is not easy to point out the most interesting games. In this system, the computer attempts to learn the user's preference on game replays based on user's explicit feedback (marking interesting points). It is not necessary to mark all the game replays and the system starts to learn from a set of subsets of replays with marks. After that, the models could predict the "likeness" on the new replays (without marks). Based on the prediction, the recommendation system could sort a number of new game replays for the user.

The replays are not stored in the form of "video" recordings. Instead, it is saved as the sequence of game events (commands or mouse clicks) by players. The replay files can be played only with original game programs (for example, StarCraft). The game simulates the game events recorded and the results should be the same with the original one because there is no randomness in the replay. In the replaying, the user's role is "Observer" and he/she can watch all the player's territory, commands, resources and units without any fog.

In the feature extraction, the raw data can be converted into a high-level one. Because the games are represented as a kind of sequential (time series) data, it is possible to get the average of values during the game and this can be used as "global features." For example, the APM (Actions per Minute) can measure the average speed of actions for the player. On the other hand, local "unit" features can be defined based on the values of specific type of units in the game. For example, the first construction time of worker units can be the local feature.

The final step is to build models for the prediction of "likeness" on replays using the training data which contain the features and the corresponding "labels." In our experiments, we

apply three types of classifiers: Decision Tree, Support Vector Machines, Bayes, and Ensemble approaches.

A. Acquisition of User's Preference

It is an interesting cultural phenomenon that gamers enjoy watching professional player's games [14]. It's possible to record all the actions of each player during the game and store them into a separate file to replay the game. This file is called as "replay file." From several gaming portals, you can find thousands of game replays. With the file, it is possible to replay the game exactly the same as the original one. During the replay, the role of the user is "Observer" and it is possible to see all the units, buildings, maps and actions of the players. It is not a kind of video recording of games but stores all the game events and restore them to generate replay.

Initially, the system requires collection of a user's explicit impression on game replays. Although it is best to get feedback implicitly, it is challenging to design such devices with high accuracy [15]. In this scenario, users watch a set of game replays and mark their interesting points in the games explicitly. It is not necessarily to identify each replay as one of "like" or "dislike." From the interview with gamers, it is easier to mark specific interesting points than to classify each game into one of two categories. They pointed out that it's quite ambiguous to say one game as clearly "preferable" or "worst."

While users watch a game replay, they mark interesting points multiple times. The number of markings is dependent on the length of the game and the user's preference. In this work, we devise a formula to normalize a user's preference on each game replays based on the markings.

$$f(x,l) = \log_{10}\left(\frac{n(x,l)+0.1}{w_l}\right) \quad (1)$$

,where the number of replays (or game videos) is $L = \{1,2,3,\dots l\}$. Let $n(x,l)$ denote the number of markings (interesting points by user) when the participant x watches the game replay l. In addition, ω_l denotes the total time of the game replay l. For example, the match is finished at 30 minutes when replay number is l=10, ω_{10} value is 30. The value of 0.1 prevents that f value by the formula $f(x,l)$ converging on zero when $n(x,l)$ is zero, that means there are no interesting scenes on replay l. The f value represents the markings frequency of replay l per seconds. We can use f value as a standard for value how much each user pays attention to the replay l regardless of differences between the replay time.

Although users do not categorize each game replay into one of "like" or "dislike" by themselves, it is necessary to label them for the supervised learning using the f value. Based on our observation, each user has different styles of markings and the distribution of f value is quite different. In this work, the categorization of game replays into one of the two or three classes is done using the f value. Each user has a different threshold value to categorize the replays according to the distribution of f value. If the problem is binary classification ("like" or "dislike"), the half of the replays with low f-value should be categorized into "dislike" and vice versa.

In this work, we formulate the classification problem as binary or a three classes (multi-class) problem. In the binary case, each sample is one of "high" and "low." "high" means that the f value is high (more than half). In the multi-class formulation, each sample is one of "low," "middle," and "high." The discretion of the class is dependent on the distribution of f value for each user. Each user has different threshold value for each class. Each class (low, middle, high) has same quantity in one user.

B. Feature Extraction

The raw-level game events are extracted from the replays (not in the form of videos) and they're analyzed to be converted as "features." Because the game is played by more than two players, the game events are a mixture from multiple players. By applying filters to the raw data, it is possible to get data from a specific player. In a two player game, you can get data from either of the players or both of them. The settings of the filtering drastically change the meaning of input signals for the next learning. This is a special property of the game replay analysis.

Each game is stored in the binary format of the StarCraft game replay. Because the game is a commercial product, the details of the format are officially unknown. However, the secrets have been exposed from some expert programmers allowed to analyze game replays and build AI programs for the game. It is possible to export all the gaming events stored in the game replays into a text file using some customized replay analyzing programs. Although there are some restrictions on the export, it is quite useful to automate the analysis of thousands of replays from the internet. The representative programs for that purpose are LMRB[2] (Lord Martin Replay Browser) and BW Chart[3]. Their tools are used not only to study research but also to analyze his/her game control by users. In fact, APM values denoted by BW Chart use the criteria to compare control ability between users.

Using the tools, it is possible to list all the gaming events (actions of players). For example, it is a sequence of records (time stamp and action type) during the game. The action includes mouse click, moving of units, construction of buildings, upgrade of specific units, and so on. Because it generates much information, it is necessary to extract useful features to reduce the dimension. In the case of a two player game, it is possible to extract gaming events just for a single player or both of them. Because our goal is to provide recommendation for users who watch the game as an "Observer" role, the gaming events are extracted ignoring the "fog-of-war" property.

In this paper, we consider two types of features (Global and Local features). Global features are extracted from the BW Chart program. They are as follows. There are fourteen features in total.

- Actions: The total number of actions (game events) by the player

[2] http://lmrb.net/
[3] http://bwchart.teamliquid.net/

- APM (Actions per Minute): It is one of important measures to guess the level of player. Usually, the professional player's APM is very high (more than 200). It means the average number of actions per minute.
- Null : The invalid actions by the player
- VAPM : It removes some invalid actions.
- APM Max, Min, and Deviation: It records the maximum, minimum APM during the game. Also, the deviation shows the stability of the player.
- Mineral and Gas: It means the total amount of minerals and gas mined in the game
- Population: The number of population produced in the game
- Units: The total number of units produced in the game
- APM Micro and Macro: It records the micro (individual) control, macro (group) control of units APM during the game.
- Game Strategy: A simple rule of sets is used to classify the game (based on the build order) into one of seven predefined strategies. For example: Bio, Two Factory, Vulture Harass, Siege Expand, Standard, Fast Drop ship, and unknown for Terran race. The rule set was defined by [4] and [11].

On the other hand, local features store the information of the first timestamp of each unit production or building construction. This feature representation has been used in other papers on strategy prediction [4],[5],[11]. For Terran race, there are 51 features that are defined for all kinds of units and buildings. Because each race has a different number of unit types, the number of features is dependent on the race. For example, Zerg race has 48 features. The local features are extracted using pre-processing.

We divided the game element of RTS into two group; the global features that represent a player's control and the whole game state (mineral, gas, units and strategy) and local features that represent time sequence of unit production and building construction of each race. As a result, if we get this information from other RTS games into log files, the suggested approach can apply to the RTS games.

Because the game is played by multiple players, it is possible to predict the user's preference based on gaming events from a single player or all players.

- Single Player View: In this setting, each sample represents the features extracted from a single player. As a result, there are two samples for a single replay if the game is a one vs. one game. For example, you can have one sample for Zerg and another sample for Terran for a Zerg vs. Terran match.
- Multiple Players View: In this setting, each sample combines all the features from the players in the replay. For example, they combine all the features from Zerg and Terran for the Zerg vs. Terran match. As a result, one replay produces only one sample.
- Single Player View + Multiple Players View: In this setting, it contains two samples from the single player view and one sample from the multiple players view. In total, there are three samples from a single replay.

In this study, we attempt to see the best combination of the features (local and global features) and the viewpoints of the game to predict the user's preference.

C. *Learning Classifiers*

From the earlier works, research reported that some classification algorithms are promising for the classification of game strategy for StarCraft [4],[5],[11]. In this work, we attempt to test several machine learning algorithms given the recommendation problem. They are as follows.

- Baseline Approaches: It is just for comparison. It predicts only one of classes. They are ZeroR and OneR [16].
- Bayes Approaches: Although this is a very simple method, the performance is surprising when tested on actual datasets. It's the probabilistic approaches based on Bayes'rule when the events are independent. We use Naïve Bayes [16].
- Support Vector Machines (SVM) Approaches: This model use the linear model to implement nonlinear class boundaries. There are many modified and expanded method based on SVM, we adopt simple SVM, using Library for SVM (LibSVM) in WEKA [16].
- Decision Tree: This model builds a visual tree that explains the decision mechanism behind the classification. In this paper, we adopt J48 [17].
- Ensemble Approaches: In this approach, it combines multiple classifiers to increase generalization ability. The key is to increase the diversity of members maximizing the synergism of them. In this work, we use the bagging [18], AdaBoostM1 [16], and Rotation Forest [19].

IV. Experimental Results and Discussion

A. *Data Samples*

In StarCraft, there are three races (Protoss (P), Terran (T), and Zerg (Z)). Each race has different types of units and buildings. The game has evolved for several years to balance the powers of the three races. There are in total six possible types of one vs. one matches. For example, they are PvP, PvT, PvZ, TvT, TvZ, and ZvZ. Usually, the matches between different races are more interesting than others. In this study, we focus on the matches between Zerg versus Terran races. From the interview with expert game players, they pointed out that the ZvT matches are usually quite fun.

In this paper, we collected replays of StarCraft from YGOSU.com. The number of replays is two hundred and all replay files are one versus one matches (Zerg vs. Terran). They

include games with different types and levels of players. Also, the length of games is highly variable from a short to long-term game (Figure 4). Game Time in Figure 4 means StarCraft time units (1 second = 24 StarCraft time units). Five undergraduate students (4 male and 1 female) were recruited to review the replays. Their task is to review the replays one by one and capture the game screen when they feel "fun or interesting". In fact, they are not novice players of the game and they have experience on the game. To minimize the fatigue on the game replay reviews, it is recommended to review 10~20 replays per day over a period of three weeks. User 1 has 5.83 ± 4.91 markings per replay and user 2 has 11.52 ± 7.87. User3 to 5 has 10.7 ± 11.64, 11.6 ± 9.7 and 9 ± 7.02, respectively.

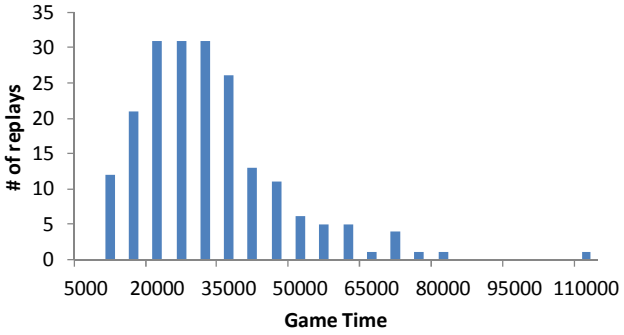

Figure 4. The distribution of replay time length

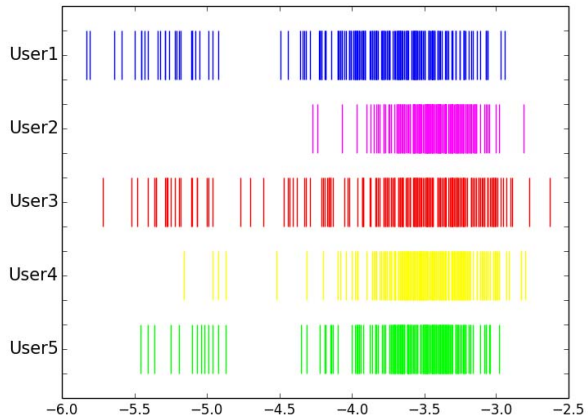

Figure 5. Distributions of f value for each participant (each bar represent one replay file)

Figure 5 shows the distribution of f value over the replays reviewed. For user 1 and 3, he shows a relatively more distributed f value pattern than users 2 and 4. It shows that each user has different styles and a range of marking numbers based on his personality. Also, each person has a different tolerance on the acceptance of markings. In this paper, the user's labels are interpreted based on the f value. It allows us to define the recommendation problem in different ways (binary or multi-class classification). It is interesting to see the performance difference depending on the number of classes. In the three classes problem, the replays are divided into three groups (33% for each category) based on the f value.

As mentioned before, we propose to use different types of features to predict the user's preference (Table I). In the design, each experimental condition has different types of features (global vs. local) and view points (single, multiple and both of them). In total, there are six different experimental conditions. It's interesting that each condition has a different number of samples available. The number of samples is dependent on the types of view point. To measure the accuracy of the machine learning algorithms, we adopt a ten-fold cross validation. WEKA[4] machine learning toolkit is used to conduct the experiments.

For the global features, it has 14 features for each player and the total number of features for the multiple view point is 28. For the local features, there are 51 features for Terran race and 48 features for Zerg race. Each race has different number of unit and building types. For the multiple player's view, it combines the features from the both players and the number of features is 99.

TABLE I
EXPERIMENTAL SETTINGS

Abbreviation	Attribute	View Point	# of dataset (# of features)
G_M	Global	Multiple	200 (28)
G_S	Global	Single	400 (14)
G_SM	Global	Single + Multiple	600 (28)
L_M	Local	Multiple	200 (99)
L_S	Local	Single	400 (99)
L_SM	Local	Single + Multiple	600 (99)

B. Results

Table II and III shows the accuracy of the different combination of binary classification using different attribute (Global vs. Local). It shows that the best accuracy is 79.16 for average of users on the binary problem with RotationForest in L_SM case. For the three classes problem, the best accuracy is 72.75 for average of users with RotationForest in L_SM case. For the result, the best approach is Local features with Single + Multiple viewpoints trained with RotationForest. In general, the local features filtered from the combined view points are better than other alternatives regardless of the number of classes. Also, the prediction accuracy is highly dependent on the choice of classification algorithms. However, the Rotation Forest is superior to other approaches.

Although the number of datasets in G_SM case is more than in G_S, some user's accuracy in G_S is higher than the accuracies in G_SM. This means that the number dataset is not too small. As shown in Table II, Naïve Bayes and SVM approach are similar with baseline classification methods; ZeroR and OneR. And the accuracies with AdaBoost have the high value in G_M and G_S case. As result in Table II, the probabilistic and SVM approaches are not suitable for global attributes. Moreover, the accuracies with global attributes show that the global attribute is not sufficient for explaining the

[4] http://www.cs.waikato.ac.nz/ml/weka/

user's mind, because there is a small gap between the best accuracy and baseline accuracy with ZeroR and OneR.

In general, the ensemble approaches (Bagging, AdaBoost, and RotationForest) are an effective solution to predict user's intention not only with global but also with local attributes. The accuracy of baseline approaches sometimes outperformed the accuracy of others. It means that the system has failed to predict user's interest. We get the best performance with respect to whole users in L_SM case, particularly the best accuracies of whole users are obtained from RotationForest classification. The RotationForest, have similar performance with Random Forest [20], is a strong ensemble learning technique to reduce learning errors and to improve diversity. In this experiment, RotationForest shows that this ensemble approach is more suitable for predicting user's preference than decision tree, bayes, and SVM approaches.

V. Conclusion and Future Work

In this work, we propose to build a personal game content recommendation system using machine learning approaches. Initially, users explicitly express his/her preference on game contents (in this paper, the game replays) and the machine learning algorithm extracts features and build models to make prediction on new game contents. It is necessary because users suffer from the huge amount of game contents produced from many competitions, tournaments and broadcasting channels. The purpose of this paper is to examine that we get successful results from some elements of prediction of user's preference and to build a personal game content recommendation using it.

In the experiments, we found that there are some different ways to extract useful features for real-time strategy game contents. For example, it is possible to extract local and global features. Also, the features are different depending on the view points of the game players. Also, the classification algorithms are one of the most important factors to build a successful recommendation system. In the experimentation, we collect two hundred replays and preferences for five users. Using the data, we tested a different combination of feature extraction approaches and machine learning algorithms. It shows that the prediction accuracy is highly dependent on the choice of features and classification algorithms. However, the local features and single + multiple view points with the RotationForest approach outperforms other alternatives on binary and multi-class problems.

Because the system is based on the user's explicit feedback, it is unavoidable to bother users in the early stage of the system. The development of an implicit feedback system is required to make the system more accessible. For example, a BCI (Brain Computer Interface) can be a promising solution for the feedback from users [21].

References

[1] G. Linden, B. Smith, and J. York, "Amazon. com recommendations: Item-to-item collaborative filtering," *Internet Computing*. IEEE, vol. 7, no. 1, pp. 76-80, 2003.

[2] T. L. Taylor, Raising the Stakes: E-sports and the Professionalization of Computer Gaming. The MIT Press, 2012.

[3] M. Buro, "Real-time strategy games: A new AI research challenge," in IJCAI, 2003, pp. 1534–1535.

[4] H. C. Cho, K. J. Kim, and S. B. Cho "Replay-based strategy prediction and build order adaptation for StarCraft AI bots," in Computational Intelligence in Games (CIG), 2013 IEEE Conference on. IEEE, pp. 1-7, 2013.

[5] H. Park, H. C. Cho, K. Lee, and K. J. Kim, "Prediction of early stage opponents strategy for StarCraft AI using scouting and machine learning," in *Proceedings of the Workshop at SIGGRAPH Asia*. ACM, pp. 7-12, 2012.

[6] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley, "Combining Search-Based Procedural Content Generation and Social Gaming in the Petalz Video Game," in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*. AAAI, pp.63-68, 2012.

[7] J. Togelius, M. Preuss, N. Beume, S. Wessing, J. Hagelbäck, G. N. Yannakakis, and C. Grappiolo, "Controllable procedural map generation via multiobjective evolution," *Genetic Programming and Evolvable Machines*, pp. 1-33, 2013.

[8] M. Kaytoue, A. Silva, L. Cerf, W. Meira Jr, and C. Raïssi, "Watch me playing, i am a professional: a first study on video game live streaming," in Proceedings of the 21st international conference companion on World Wide Web, 2012, pp. 1181–1188.

[9] S. Ontanón, et al. "A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 5, no. 4, pp.293-311, 2013.

[10] S. Liu, S. J. Louis, and M. Nicolescu, "Comparing heuristic search methods for finding effective group behaviors in RTS game," in Evolutionary Computation (CEC), 2013 IEEE Congress on, 2013, pp. 1371–1378.

[11] B. G. Weber, and M. Mateas, "A data mining approach to strategy prediction," in *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pp. 140-147, 2009.

[12] G. Synnaeve, and P. Bessiere, "A bayesian model for opening prediction in rts games with application to starcraft," in *Computational Intelligence and Games (CIG), 2011 IEEE Conference on*. IEEE, pp. 281-288, 2011.

[13] G. Synnaeve, and P. Bessiere, "A Bayesian Model for Plan Recognition in RTS Games Applied to StarCraft," in *Eighth Artificial Intelligence and Interactive Digital Entertainment Conference*. AAAI, pp. 79-84, 2011.

[14] G. Cheung, and J. Huang, "StarCraft from the stands: Understanding the game spectator," *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 763-772, 2011.

[15] G. N. Yannakakis, and J. Togelius, "Experience-driven procedural content generation," *IEEE Trans. on Affective Computing*, vol. 2, no. 3, pp. 147-161, 2011.

[16] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques. Elsevier, 2011.

[17] J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993.

[18] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123-140, 1996.

[19] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1619-1630, 2006.

[20] L. Breiman, "Random Forests," *Machine learning*, vol. 45, no. 1, pp. 5-32, 2001.

[21] B. van de Laar, H. Gurkok, D. Plass-Oude Bos, M. Poel, and A. Nijholt, "Experiencing BCI Control in a Popular Computer Game" *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 5, no. 2, pp. 176-184, 2013.

TABLE II
ACCURACY OF BINARY CLASSIFICATION USING GLOBAL ATTRIBUTE

Case Name	ZeroR	OneR	J48	NB	LSVM	Bagging	AdaBoost	RF	Avg±St. Dev.
G_M(U1)	49.5	48.15	52.2	49.95	50.1	55.45	**59.5**	56.85	**56.85±52.71**
G_M(U2)	49.5	52.4	50.45	49.65	50.1	50.6	53.05	**54.15**	54.15±51.24
G_M(U3)	49.5	50.35	51.2	51.25	49.9	**55.9**	53.4	53.75	53.75±51.91
G_M(U4)	49.5	50.2	54.25	48.85	49.9	49.8	**55.45**	53.75	53.75±51.46
G_M(U5)	49.5	51.3	49.55	46.75	50.2	**52.4**	**52.4**	49.8	49.8±50.24
Avg±St. Dev.	49.5±0.0	50.48±1.47	51.53±1.62	49.29±1.49	50.04±0.12	52.83±2.47	**54.76±2.58**	53.66±2.25	
G_S(U1)	50.5	51.43	56.13	50.4	50.5	**58.45**	53.15	56.13	53.34±2.96
G_S(U2)	50.5	49.45	51.58	51	50.5	50.95	**57.13**	52.63	51.72±2.22
G_S(U3)	50.5	56.33	53.73	50.6	50.5	51.78	**59.03**	57.73	**53.78±3.27**
G_S(U4)	50.5	50.5	51.28	50.13	50.5	52.5	**56.65**	56.23	52.29±2.5
G_S(U5)	50.5	**56.73**	52.93	49.15	50.5	49.93	53.48	54.78	52.25±2.49
Avg±St. Dev.	50.5±0.0	52.89±3.04	53.13±1.74	50.26±0.62	50.5±0.0	52.72±2.99	**55.89±2.25**	55.5±1.71	
G_SM(U1)	50.5	49.27	54.03	51.45	50.5	57.28	52.82	**57.95**	**52.98±3.02**
G_SM(U2)	50.5	47.78	50.78	49.12	50.5	51.52	**57.03**	54.68	51.49±2.8
G_SM(U3)	50.5	52.22	53.33	52.43	50.5	52.5	56.58	**56.73**	53.1±2.25
G_SM(U4)	50.5	50.05	53.33	48.43	50.5	50.98	**56.7**	56.15	52.08±2.81
G_SM(U5)	50.5	50.87	51.35	49.62	50.5	51.32	50.5	**54.67**	51.17±1.42
Avg±St. Dev.	50.5±0.0	50.04±1.49	52.56±1.26	50.21±1.49	50.5±0.0	52.72±2.34	54.73±2.61	**56.04±1.25**	

TABLE III
ACCURACY OF BINARY CLASSIFICATION USING LOCAL ATTRIBUTE

Case Name	ZeroR	OneR	J48	NB	LSVM	Bagging	AdaBoost	RF	Avg±St. Dev.
L_M(U1)	49.5	58.45	56	58.65	50.1	58	**58.7**	56.9	**55.79±3.57**
L_M(U2)	49.5	46.85	49.55	47.1	**50.1**	49.65	46.9	48.1	48.47±1.29
L_M(U3)	49.5	**57.6**	51.5	57.05	49.9	55	50.75	54.95	53.28±3.04
L_M(U4)	49.5	51	52.45	50.7	49.9	54.7	56.2	**56.45**	52.61±2.63
L_M(U5)	49.5	48.8	53.35	**57.95**	50.2	54.55	51.65	54.05	52.51±2.87
Avg±St. Dev.	49.5±0.0	52.54±4.67	52.57±2.13	54.29±4.57	50.04±0.12	**54.38±2.68**	52.84±4.16	54.09±3.17	
L_S(U1)	50.5	50.4	53.73	51.98	50.5	**57.4**	50.93	56.98	**52.8±2.74**
L_S(U2)	**50.5**	47.4	49.13	46.08	**50.5**	48.18	48.2	47.2	48.4±1.47
L_S(U3)	50.5	51.33	**55.2**	51.5	50.5	52.93	51.85	54.23	52.26±1.61
L_S(U4)	50.5	50.08	53.78	49	50.5	55.7	**58**	54.58	52.77±3.0
L_S(U5)	50.5	45.95	52.5	52.15	50.5	53.85	52.18	**54.45**	51.51±2.47
Avg±St. Dev.	50.5±0.0	49.03±2.02	52.87±2.06	50.14±2.33	50.5±0.0	**53.61±3.12**	52.23±3.21	53.49±3.3	
L_SM(U1)	50.5	59.08	70.78	56.85	50.8	74.55	60.68	**79.05**	62.79±10.09
L_SM(U2)	50.5	57.47	71.78	52.27	50.83	70.83	54.95	**78.62**	60.91±10.38
L_SM(U3)	50.5	66.62	72.43	55.48	50.83	72.73	56.88	**79.98**	**63.18±10.51**
L_SM(U4)	50.5	60.13	72.77	55.65	50.5	73.8	62.2	**79.5**	63.13±10.37
L_SM(U5)	50.5	59.92	73.35	57.32	50.8	72.4	57.42	**78.63**	62.54±10.1
Avg±St. Dev.	50.5±0.0	60.64±3.13	72.22±0.88	55.51±1.77	50.75±0.13	72.86±1.27	58.43±2.64	**79.16±0.52**	

The accuracies of eight models: NB=NaiveBayes; LSVM=LibSVM; RF=RotationForest
U1,U2,U3,U4,U5 means User 1,User 2, User 3, User 4, and User 5 respectively.