Semisupervised Game Player Categorization From Very Big Behavior Log Data

Jing Fan, Shaowen Gao, Yong Liu, *Member, IEEE*, Xinqiang Ma, Jiandang Yang, and Changjie Fan

***Abstract*—Extracting the specific category of the players, such as the malignant Bot, from the huge log data of the massive multiplayer online role playing games, denoted as MMORPGs, is an important basic task in game security and personal recommendation. In this article, we propose a parallel semisupervised framework to categorize specific game players with a few label-known target samples, which are denoted as bait players. Our approach first presents a feature representation model based on the players' level granularity, which can acquire aligned feature representations in the lower dimensional space from the players' original action sequences. Then, we propose a semisupervised clustering method, extended from the bisecting *k*-means model, to extract the specified players with the help of those bait players. Due to massive amounts of game log data, the computation complexity is an extreme challenge to implement our feature representation and semisupervised extraction approaches. We also propose a hierarchical parallelism framework, which allows the data to be computed horizontally and vertically simultaneously and enables varied parallel combinations for the steps of our semisupervised categorization approach. The comparable experiments on real-world MMORPGs' log data, containing more than 465 Gbytes and million players, are carried out to demonstrate the effectiveness and efficiency of our proposed approach compared with the state-of-the-art methods.**

***Index Terms*—Feature representation, game behavior mining, parallel processing, semisupervised learning.**

I. Introduction

CATEGORIZING game players from very big log data is an important task in gamer behaviors mining and can be widely used in malignant Bot detection [1]–[6] and personalized recommendation [7], [8], etc. In this article, we will focus on the parallel player categorization on massive multiplayer online role playing games (MMORPGs). The MMORPGs allow users to play their favorite roles, to acquire different skills, and free to act with the game settings. Unlike single-player games, MMORPGs provide a lot of freedom and interaction with other players in the virtual world.

As there are millions of game players in the MMORPGs, those players can act almost all the social behaviors in the virtual game environment, and the game system will record all their sequential actions in the log, their behavior logs will accumulate quickly into a very big data set. Then, the problem of specific players extraction will face the following three challenges.

The first challenge is how to efficiently construct the unified representation model on those very big variational event logs. The raw game logs are stored as a long sequence of players' actions, and those unstructured data cannot be applied with classification or clustering methods directly. Then, the structured feature representation model for those raw logs becomes the critical task in the player categorization, and at the same time, the structured feature representation model also requires efficiently computing due to the huge number of logs.

The second challenge is how to categorize the tiny set of specific players from very big imbalanced and imperfect data set. The target category of players normally only accounts for quite a small proportion of the overall players [9], and in our study, the company of NetEase Games discovered less 0.5% of the all players were the malignant Bot players in MMORPGs of Fantasy Westward Journey Online. From the viewpoint of binary classification, it can be regarded as a typical imbalance classification problem, however, the characteristics of those specific category of the players will shift with the game updating or intentional hiding,[1] which will lead to the pretrained classification model fails to recognize those players of the specific category. At the same time, when categorizing a specific category of the players, we will manually find a small portion of the players belonging to that category, and the characteristics of these manually labeled players are useful to the player categorization. Although the labeled data are limited for the large number of players, it should also be specially considered during the categorization. For example, NetEase Games' operation teams can label some types of game bots, these labeled samples provided very useful information, such as action count, activity time, and chat time, to distinguish the player categorization that game bots are belonging to and human players. The small part of the labeled game bots are

Manuscript received October 12, 2020; accepted February 21, 2021. Date of publication April 23, 2021; date of current version May 18, 2022. This work was supported in part by the Grants from key research and development plan Program of Zhejiang Province under Grant 2019C01004 and key research and development plan program of Guangdong Province under Grant 2019B010120001. *(Jing Fan and Shaowen Gao are co-first authors.)* This article was recommended by Associate Editor L. Wang. *(Corresponding author: Yong Liu.)*

Jing Fan and Shaowen Gao are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China (e-mail: fanjing105@zju.edu.com; gao_shaowen@126.com).

Yong Liu is with the State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China (e-mail: yongliu@iipc.zju.edu.cn).

Xinqiang Ma is with the Department of Software, Chongqing University of Arts and Sciences, Chongqing 402160, China (e-mail: xinqma@163.com).

Jiandang Yang is with the College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China.

Changjie Fan is with Fuxi Tech, Netease Corporation, Hangzhou 310052, China (e-mail: fanchangjie@corp.netease.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2021.3066545.

Digital Object Identifier 10.1109/TSMC.2021.3066545

[1]The malignant Bot players will random change their action modes to hide themselves.

very important for automatic detecting the bot categorization by machine learning methods.

The third challenge is how to optimize the categorization approach into the scalable distribute computation architecture. In real-word applications, such as bot detection in game companies, the solutions with a high-available parallel computation framework are needed to deal with massive log data. It also requires the categorization approach, which can be easily executed in parallel and utilize the parallel computation resources thoroughly by a proper optimization of the parallelism.

Motivated by the above challenges, we propose a parallel semisupervised player categorization approach, which consists of a structured feature model and its corresponding constructing method on the log sequence, a semisupervised player extraction algorithm from the structured feature model and a hierarchical parallel strategy based on varied granularity, which can achieve optimal parallel computing for feature constructing and player extraction. The detailed contributions are summarized as follows.

1) We propose a formal unified feature representation model, which transform the unaligned player action sequences into fixed-length vectors that are used in a ranking-based feature selection method. Instead of the classical approach on sequence data, such as the hidden Markov model (HMM), which mainly models on variational length series, we aim to construct fixed-length sequences based on the granularity of the player's level. Furthermore, we propose a normalization approach to eliminate the event frequency bias of different levels in MMORPGs. To avoid the curse of the dimension, we incorporate feature selection to find the beneficial features for extracting the target players. Our proposed feature representation model has been empirically proved to be a unified feature representation in different categorization problems in MMORPGs.
2) To extract specific players with incomplete labels, a bait-conducted bisecting clustering (BBC) method is proposed to retrieve a certain category of players using a very few samples of those players. As mentioned before, the raw data set is mixed with the target category and other unknown categories, and our method can take full use of the known sparse labels of the target category. Instead of training the specific classifier to determine several certain categories of players, our bait-conducted extracting method can extract different categories of players only by changing the bait player set. Our approach is able to achieve superior performance compared with other state-of-the-arts methods, which can be proved by the real experiments in MMORPGs.
3) Our approach is fully optimized on the Spark architecture. A hierarchical parallel strategy based on varied granularity is proposed to improve the row parallel into a row–column parallel, which can support varied parallelism on different stage of our semisupervised categorization approach. This can best utilize the parallel computation resources. The comparable experimental results in real MMORPGs demonstrate that our proposed hierarchical parallel strategy can perform much better than the parallelism of the original Spark.

The following sections in this article are organized as follows. In Section II, the related works are given. In Section III-B, the feature representation method is formally given. Section III-C presents the specific player extraction algorithm. The hierarchical parallel strategy for our semisupervised categorization approach is proposed in Section III-D. The experiments in real MMORPGs are discussed in Section IV to evaluate the validation of our approach. Then, we give a conclusion of the proposed work in Section V.

II. Related Works

The challenges as mentioned in the previous section consist in the three main steps for the player categorization: 1) the structured representation of game features; 2) extraction algorithm; and 3) parallel optimization for massive data. This section will introduce the related works on these three fields, respectively.

To model the player behaviors in MMORPGs, researchers have proposed varied structured feature representation models from raw logs. Based on the elemental events, the intuitive and simplest feature representation model is proposed by statistic histogram of the specific player actions [3]. However, this intuitive feature representation model may express the players' behaviors on a too small granularity (there are almost 10 000 kinds of events logged in the MMORPGs), thus it can not well distinguish different categories of players in such a fractional feature granularity. Then, Chung *et al.* [1], [5] further summarized the player behaviors into more rough and superior categories such as combat, collection, and movement, quantified by accumulative action frequency within a common duration. In their method, the features mainly involved in counting the high-level actions' frequencies in a certain period for analysis, which may be lack of the time series information hiding in the actions. Platzer [6] employed the edit distance to compute the similarity between combat sequences. However, their feature representation model concerned the game events partially and did not take the incomplete labels into account, which made it hard to be generalized to the other category of players. In this article, we try to propose a novel feature representation model in a proper feature granularity, which can take full use of the information contained in the raw logs and different communities of players.

The above feature representation models are mostly used in Bot detection, which tries to differ malignant program Bot players [6][2] from human players by the massive log data. Bot detection can be regarded as a specific problem of the player communities categorization, and many extraction algorithms have been introduced. Unsupervised methods were employed to find out how the users form distinctive communities [10]–[12]. Thurau and Bauckhage [12] considered the MMORPGs avatars' level-up and employed matrix factorization method to obtain the lower dimensional data embedding as the interpretable group categorizations. Drachen *et al.* [11]

[2]Bot here refers to a program that allows a user to play the game partially or completely unattended.

processed the high-dimension player behaviors with k-means and SVM to extract both general and extreme behaviors. Bauckhage *et al.* [10] further compared the performance of typical clustering algorithms, including k-means, matrix factorization, and spectral clustering on player communities categorization. Due to the connectedness among the community members, graph grew to be an effective approach to identify different communities [13], [14].

The semisupervised learning is also widely used in community detection problems, which usually assumes a few known data labels and predefined structure of the data, including cluster assumption, manifold assumption and local and global consistency assumption. Local and global consistency assumption extended the other two assumptions, represented by the algorithms such as label propagation (LP) [15] and label spreading [16]. The nearby points are defined as local and points with the same structure are defined as global. These methods can be categorized into graph-based algorithms, which constructed a similarity graph over all observations to explore community structures from real networks [17], [18]. Then, some developed works were introduced for complex networks. In [19], the original labels are propagated to using synthesized linear neighborhood with sufficient smoothness. To enhance the supervised information, Zhang *et al.* [20] employed special LP to create and enrich the pairwise constraints. For game player categorization in this article, only specified group will be extracted with the partially labeled players. Hence, we extend semisupervised learning into our framework due to the incomplete labels. Unlike the previous methods that aimed to mine all the groups, we only focus on the interested group, such as malignant Bots, which is distinguished from traditional community detection.

The log data are usually characterized by large volumes and high dimensions and, thus, the model should be implemented in a parallel processing framework and optimized on the parallelism. Apache Spark evolving from MapReduce [21], [22], a full-stack parallel processing framework for big data, is employed as the proper parallel processing framework for our methods. The Spark features in-memory computation and directed acyclic graph (DAG) scheduler, which promises multiplied speed-up performance over MapReduce and highly simplified job pipeline. The resilient distributed datasets (RDD) determines the parallel abstraction of distributed data in Spark. However, it does not support the nested structure, which means that only one single layer of parallelism is available. To optimize Spark out of its intrinsic capability, many works proposed an extra data segmentation on various models. Splitting the search-space is the most direct way for further parallelism [23], [24]. Bharill *et al.* [23] partitioned the big data into various chunks when training fuzzy clustering and processing different chunks with higher parallelism. Savage *et al.* [24] proposed a segmentation plan on mining contrast patterns, which is hard to be modeled in a parallel manner, especially for large volume and high dimensional data set. They implemented the model on Spark by splitting and merging the subtrees recursively, which can achieve a highly scalable and effective parallelism. Besides the sample-unit partition, parallelizing the features is also incorporated. In the work of [25], Chen *et al.* proposed vertical data partitioning in parallel random forest algorithm. It can avoid unnecessary transmission on the partial training set on the distinctive tree and get the network transmission cost optimized by partitioning the features into feature subset. Inspired by the vertical partitioning, we propose a hierarchical parallel framework utilizing both Spark and Scala parallel strategies and, thus, the data can be computed in horizontal and vertical parallelism at the same time.

III. Our Semisupervised Categorization Approach

Our semisupervised categorization approach consists of three parts: 1) the fixed-length feature representation; 2) player categorization algorithm; and 3) hierarchical parallelism framework (HPF). The raw logs can be assembled in the sequences, however, these unaligned sequences cannot be fed to learning algorithms directly. We propose an approach to construct fixed-length vectors from the log data. To categorize the players who we are interested in, we design a semisupervised algorithm inherited from bisecting k-means [26], to extract the whole group utilizing the known part of players. To optimize the running cost in the feature representation step, we also propose an HPF on Apache Spark, combined with Scala parallel collection.

A. Problem Formulation

To mine the behaviors and communities from the huge volume of raw game logs, we first consider to represent players with the structured features. The player's events logged in MMORPGs are usually triggered by players or the game system. The events related to players can be divided into two types: 1) initiative and 2) passive. For example, killing a monster is an initiative event while upgrading himself is a passive event, since "killing" is directly performed by the player while "level-up" is triggered by the system. There are also system events which are related to built various scenes in the game, such as dropping items. As those system events are only controlled by game system, we mainly focus on the player-related events in the player categorization task.

To formulate the concepts, we give the formal definitions as follows.

Event is denoted by a quadruple $e =< r, a, o, t >$, which means a player r performs an action a over the object o at the timestamp t.

Action sequence is the aggregation of sequenced events for each player. We denote the action sequence of user u as $S_u = e_1 e_2 e_3 \ldots e_N$. e_i is the ith event, $e_i(t)$ is the timestamp of e_i, for $e_i \in S_u$, $1 \leq i \leq N$, it satisfies $e_i(r) = u$ and $e_i(t) \leq e_{i+1}(t)$. It needs to be noticed that the sequence size N is different for each player.

MMORPGs granularity is defined as a proper split scope to align the action sequences of different players. The size of S_u keeps increasing when the player spends more time on the game. Moreover, different players have different size of S_u. It is hard to align different players directly from those variational-length sequences. Thus, we need to define a proper MMORPGs time granularity for constructing feature vectors.

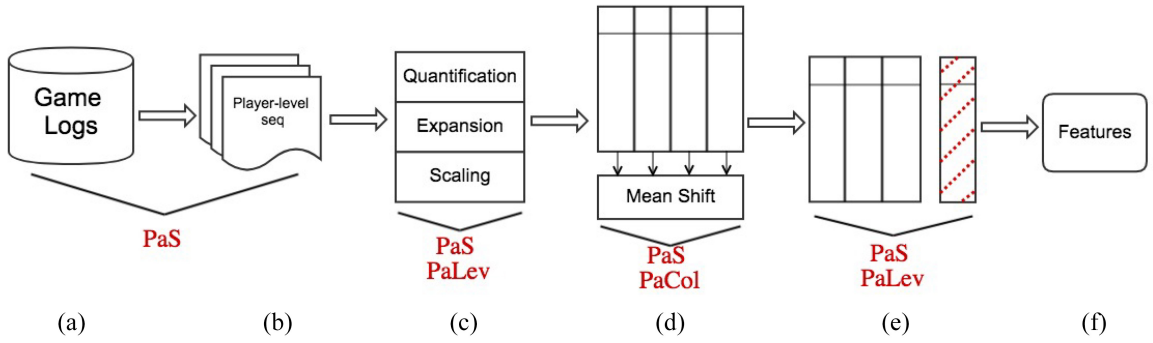


Fig. 1. Feature representation steps and the corresponding parallelism used in each step. (a) Original game logs. (b) Player-level sequences. (c) Quantification for action frequency, union expansion for level sequence and scaling. (d) and (e) One-dimension data to be clustered. (f) Top-k features are selected.

Basically speaking, players in MMORPGs upgrade by levels if they accumulate enough experience points in the game. So, we employ the level as the time granularity because it is an internal granularity division shared among all players. In terms of other granularity, it is hard to get all action sequences aligned. So, the upgrade events are chosen to split the growth of player u into m parts, $S_u = L_1 L_2 \dots L_m$, here, level sequence $L_k = e_1^k e_2^k \dots e_n^k$, it also satisfies $e_i^k(r) = u$ and $e_i^k(t) \le e_{i+1}^k(t)$, $1 \le k \le m$, $1 \le i \le n$. Note that the sequence length of S_u is variational from player to player, while the number of levels m can be a global parameter for all players as we expect. Selected with the same level range, the action sequences can be aligned with each other in the granularity of the level.

Level Union: The distinct events set of player u in level k is defined as $E_u^k = \{\text{distinct}(e_i^k(a)) \,|\, \forall e_i^k \in L_k \quad \& \quad e_i^k(r) = u\}$. Given level k, the level union can be modeled as $E^k = E_{u_1}^k \bigcup E_{u_2}^k \bigcup E_{u_3}^k \cdots \bigcup E_{u_p}^k$, where $p = |U|$ is the number of players. E^k is the event union of all players in the same level, and all m levels will be processed by the operator of the level union. Then, for player u, his level events are described as $\{E^1, E^2 \dots E^m\}$, compared to the former raw events $\{E_u^1, E_u^2 \dots E_u^m\}$.

Bait players, which are labeled manually, represent the known examples of the target category and, usually, there are very few bait players in the real case. They are utilized to induce the unknown players of the same category and this is a semisupervised problem. This also can be considered as an incomplete label problem, where the labeled players belong to the target category while the unlabeled players include both target category and other categories. Distinguished from other semisupervised models, it is unnecessary to label bait players for each category, while only those of the target category are essential. Actually, how many categories is very hard be known in practice, especially in the game. A specific player category G ($G \subseteq U$) shares similar playing style, a few instances of category G are labeled manually as G_{bait}, $G_{\text{bait}} \subseteq G$, $|G_{\text{bait}}| \ll |G|$. Our purpose is to extract all the members of G with G_{bait}.

B. Feature Representation from Variational Big Data

Fig. 1 presents the process of our feature representation approach. We process unstructured log data in step (a) and step (b). Let D represent the raw log data, $D = \{e_1, e_2, e_3 \dots\}$. In step (a) and step (b) of Fig. 1, the player set U is retrieved from D, $U = \{\text{distinct}(e(r)) | e \in D\}$, where $\text{distinct}(e(r))$ denotes the nonrepetitive players in the data set. For each player $u \in U$, his action sequence can be constructed as $S_u = e_1 e_2 e_3 \dots e_N$. By level granularity, S_u is discretized to $S_u = L_1 L_2 \dots L_m$. In step (c), we carry out quantification, expansion, and scaling operations, to transform categorical event features into numeric frequency, align the sequences and remove the growth biases. Steps (d) and (e) are the process of feature selection by events clustering. In step (f), the postprocessed features are attained eventually.

1) Quantification for Action Frequency: Quantification in Fig. 1(c) aims to quantify the event sequences generated from raw log data. In MMORPGs, event (action) frequency is a proper quantitative feature applied to infer the behavior pattern of players [2]–[5], so we choose frequency as our quantification measurement. Quantification is performed at level granularity. E_u^k denotes the distinct events set of player u in level k. Assuming $|E_u^k| = l$, E_u^k can also be denoted as $\{a_1^k, a_2^k \dots a_l^k\}$. Then, we can count the frequency of each action in E_u^k for player u in level k, which is $C_{L_k} = ((a_1^k, c_1^k), (a_2^k, c_2^k), \dots, (a_l^k, c_l^k))$. c_i^k denotes the frequency of action a_i^k for player u, and it can be calculated as follows:

$$c_i^k = \text{count}\left(a_i^k\right) \quad \forall e \in L_k \,\&\, e(a) = a_i^k, e(r) = u, 1 \le i \le l.$$

Similar to E_u^k, C_{L_k} varies with users and levels. E.g., the action sequence of player u_1 in level 1 is $L_1 = BDDAAADAA$, so his events set is $E_{u_1}^1 = \{A, B, D\}$, and the quantified action set is $C_{L_1} = ((A, 5), (B, 1), (D, 3))$.

2) Union Expansion for Level Sequence: Given two players u_1 and u_2 of level 1, their quantified sequences may be different, which are $C_{L_1}(u_1) = ((A, 5), (B, 1), (D, 3))$ and $C_{L_1}(u_2) = ((F, 5), (A, 1))$. Obviously, neither the events nor the amounts are aligned. Though we have acquired numeric features, they still keep categorical features that are actions $e_k^i(a)$. We expanded C_{L_k} to $C'_{L_k} = ((a_1^k, c_1^k), (a_2^k, c_2^k), \dots, (a_s^k, c_s^k))$ by the level union, where $s = |E^k|$ is expanded from l and shared with all players. Compared with the unaligned events in C_{L_k}, C'_{L_k} is consistent for all players of level k. For the action $a \in E^k$, $a \notin E_u^k$, the frequency is set to 0. Each player is performed in this way

in level k, until all levels are expanded. By level union, quantified action sets of all players get aligned. We simplify the expanded sequence to $N_{L_k} = \{c_1^k, c_2^k, \ldots, c_s^k\}$. $q = \sum_{i=1}^{m} s_i$ refers to the number of total features. Assembling the consequent levels to constitute the numerical features of player u, we have entirely aligned features $F_u = N_{L_1} N_{L_2} \ldots N_{L_m}$. Take the examples before, now we have level union $E^1 = (A, B, D, F)$ for u_1 and u_2 in level 1, the quantified action sets are expanded to $C'_{L_1}(u_1) = ((A, 5), (B, 1), (D, 3), (F, 0))$ and $C'_{L_1}(u_2) = ((A, 1), (B, 0), (D, 0), (F, 5))$. After the categorical event features are removed, we have aligned numeric features $N_{L_1}(u_1) = (5, 1, 3, 0)$ and $N_{L_1}(u_2) = (1, 0, 0, 5)$.

3) Ranking-Based Feature Selection: The features generated by the level union are likely to raise the problems of sparsity and high dimensionality, which may also cause the irrelevance and redundancy [27]. Moreover, some studies [10], [28]–[30] have proved that high-dimensional data must be dimensionally reduced before applying metric-based clustering such as k-means. The features in high-dimensional space might be equally far apart as the assignment step is possibly influenced by minuscule random variation, which will cause the definition of similarity failed [10]. Therefore, we consider applying feature selection to reserve top-k features in a lower dimensional space.

More specifically, we aim to find an optimal feature subset that has superior ability of discriminating the group of bait players from other players. In the first step of feature selection, each column is as one-dimension data to be clustered, so that each feature in the feature space is individually evaluated. Here, we select mean shift [31] as the clustering model, which is a nonparametric method and does not require a predefined distribution. It seeks all possible patterns in the data, without manual assignment of K. The Gaussian kernel is selected to map the distances.

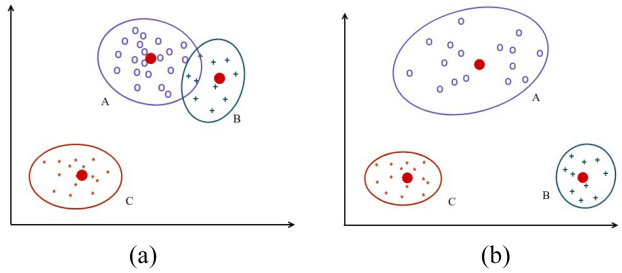
After applying one-dimension clustering for each feature, we further propose the evaluation metrics to achieve feature selection based on level granularity, which considers both supervised and unsupervised measures to evaluate the importance of features.

Evaluation Metrics: Let $\{P_1, P_2 \ldots P_k\}$ represent the clusters after applying one-dimension clustering on the feature C. Similar to the $F1$ score, we present the metrics with respect to G_{bait} for each clusters, and it is denoted as F^i_{bait}

$$F^i_{\text{bait}} = \frac{2 \cdot R^i_{\text{bait}} \cdot P^i_{\text{bait}}}{R^i_{\text{bait}} + P^i_{\text{bait}}}$$
$$R^i_{\text{bait}} = \frac{m^i_{\text{bait}}}{\sum_{j=1}^{k} m^j_{\text{bait}}}, \qquad P^i_{\text{bait}} = \frac{m^i_{\text{bait}}}{m^i} \tag{1}$$

where m^i and m^i_{bait} are the number of all players and number of labeled bait players in cluster P_i. F^i_{bait} aims to evaluate each cluster with the help of bait players. Thus, it can be seen as a supervised manner.

In our approach, the cluster P_t in $\{P_1, P_2 \ldots P_k\}$ will be selected as the best cluster on feature C if P_t has the highest evaluation metrics F^i_{bait}. F^t_{bait} of P_t is extended to be one kind of evaluation of feature C, we called it the $F1$ score of feature C. In order to balance the influence of distribution randomness



(a) (b)

Fig. 2. Two cases of the cluster distribution. (a) Good cohesion but bad separation. (b) Good separation but bad cohesion.

of bait samples, we also propose unsupervised metrics for each feature, the feature C's partial sums of squared error (PSSE) and partial sum of squares between (PSSB) are defined as follows:

$$\text{PSSE} = \frac{1}{m^t} \sum_{x \in P_t} \text{dist}(x, c_t) \tag{2}$$
$$\text{PSSB} = \text{dist}(c, c_t) \tag{3}$$

where c_t denotes the center of cluster P_t, c denotes the center of all clusters on the same feature, and $\text{dist}(a, b)$ denotes the Euclidean distance function between a and b. For feature C_i, its evaluation metrics can be presented by the triple $(F_i, \text{PSSE}_i, \text{PSSB}_i)$, $i = 1, 2 \ldots q$. Here, q is the number of features in a level, F_i, PSSE_i, and PSSB_i denote the $F1$ score, PSSE and PSSB of feature C_i, respectively.

The metrics should be normalized within the same level, as the feature selection is processed based on level granularity. We normalize all kinds of evaluation metrics in the same level by min–max normalization. The metrics triple is scaled to [0, 1] as follows:

$$F'_i = \frac{F_i - \min(F_k)}{\max(F_k) - \min(F_k)} \tag{4}$$
$$\text{PSSE}'_i = \frac{\max(\text{PSSE}_k) - \text{PSSE}_i}{\max(\text{PSSE}_k) - \min(\text{PSSE}_k)} \tag{5}$$
$$\text{PSSB}'_i = \frac{\text{PSSB}_i - \min(\text{PSSB}_k)}{\max(\text{PSSB}_k) - \min(\text{PSSB}_k)}. \tag{6}$$

For feature C_i in level k, F_i is the $F1$ score of feature C_i, $\min(F_k)$ is the minimum $F1$ score of all features in level k, and $\max(F_k)$ is the maximal $F1$ score of all features in level k. Unlike PSSB, PSSE is inversely related with the quality of clusters, thus it is compared to the max value for the positive correlation in (5).

Then, the score of feature C_i can be calculated as

$$\text{score}_i = F'_i + \text{PSSE}'_i * \text{PSSB}'_i. \tag{7}$$

The score consists of two parts, supervised F'_i and unsupervised $\text{PSSE}'_i * \text{PSSB}'_i$. As they are all normalized between 0 and 1, the final score will be varied in [0, 2].

$\text{PSSE}'_i * \text{PSSB}'_i$ depends on both cohesion and separation, making score_i directly proportional to the clustering performance. Fig. 2 shows two cases of the cluster distribution. As Fig. 2(a) shows, the points in cluster A are cohesive enough but too close to be separated from B. In Fig. 2(b), cluster A is separated far from other clusters but it is not cohesive enough. Using our scoring model, these two conditions will get low scores.

Algorithm 1 Ranking-Based Feature Selection

Input: Player data set $\{C_1, C_2 \dots C_q\}$ with q feature in m levels, N_{top};
Output: Selected $N_{\text{top}} \cdot m$ features;
1: **for** each C_i **do**
2: Clustering column C_i into $\{P_1, P_2 \dots P_k\}$ with mean shift model;
3: Compute F_i, PSSE_i and PSSB_i of C_i;
4: **end for**
5: $SF = \emptyset$
6: **for** each level j, $j \in \{1, 2, ..., m\}$ **do**
7: **for** each column C_i in level j **do**
8: Normalize to obtain $F_i^{'}$, $\text{PSSE}_i^{'}$, $\text{PSSB}_i^{'}$;
9: Compute $\text{score}_i = F_i^{'} + \text{PSSE}_i^{'} \cdot \text{PSSB}_i^{'}$;
10: **end for**
11: Select the largest N_{top} columns ranking by score_i in j, denoted as $\{C_j^1, C_j^2 \dots C_j^{N_{\text{top}}}\}$;
12: $SF = SF \bigcup \{C_j^1, C_j^2 \dots C_j^{N_{top}}\}$
13: **end for**
14: **return** SF.

Ranking-Based Feature Selection by Level: Given the parameter N_{top}, the largest N_{top} features ranked with score_i will be selected in each level. Thus, the number of dimensions is greatly reduced from $q \cdot m$ to $N_{\text{top}} \cdot m$. The selected features are input to the following extraction algorithm. The feature selection process is also presented in Algorithm 1.

C. Semisupervised Players Extraction Algorithm

After applying the ranking-based feature selection, we then present the BBC algorithm for semisupervised categorization, guiding by the bait player set G_{bait}. The BBC will extract those players, who are similar to the players in bait player set G_{bait}, from the entire player set U. These extracted players are target players. In our solution, we employ the clustering method that uses incomplete player labels as an important clue, instead of classifying methods that may suffer from class-imbalanced problem due to the partially labeled players. Our approach is based on the assumption that if the algorithm can successfully extract those target players from U, players in G_{bait} should be included in the set of target players and not be included in other parts. Thus, the bait players can be utilized to conduct the process of clustering and distinguish the target player cluster. Our solution can be regarded as a semisupervised clustering approach.

1) Bait-Conducted Bisecting Clustering Algorithm: Our approach is presented in Algorithm 2, which extracts the target category using a bisecting k-means [26] based divisive hierarchical clustering. There are two main steps: step I is splitting the set into two clusters by the basic k-means algorithm, and step II is choosing the optimal cluster to be split in the next iteration. In step II, the recall R_{bait} of the bait set is employed to select the optimal cluster. Our approach allows a portion of samples to be removed iteratively and the refined subset will go on approaching the target group.

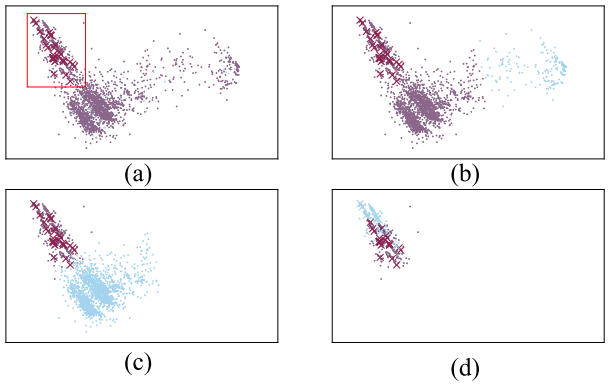


Fig. 3. BBC process.

Algorithm 2 BBC Algorithm

Input: Players data set U and G_{bait} after feature selection, $\Delta S_{\min}$, $\Delta R_{\max}$;
Output: The player set G belonging to the same category of G_{bait};
1: $R_{\text{last}} \leftarrow 1$, $\Delta S \leftarrow 1$, $\Delta R \leftarrow 0$;
2: **while** $\Delta S > \Delta S_{\min}$ **and** $\Delta R < \Delta R_{\max}$ **do**
3: Bisect U into U_1, U_2 with bisecting clustering;
4: $R_{\text{bait}} \leftarrow \max(\frac{\|G_{\text{bait}} \cap U_1\|}{\|G_{\text{bait}}\|}, \frac{\|G_{\text{bait}} \cap U_2\|}{\|G_{\text{bait}}\|})$;
5: U_{best} is set to the cluster corresponding to R_{bait};
6: $\Delta S \leftarrow \frac{\|U - U_{\text{best}}\|}{\|U\|}$;
7: $\Delta R \leftarrow R_{\text{last}} - R_{\text{bait}}$;
8: $R_{\text{last}} \leftarrow R_{\text{bait}}$;
9: $G \leftarrow U$;
10: $U \leftarrow U_{\text{best}}$;
11: **end while**
12: **return** G.

$\Delta S_{\min}$ and $\Delta R_{\max}$ are input thresholds for the minimal variation of the cluster size and the maximal variation of the recall on bait set. ΔS and ΔR are used to store the variations of the cluster size and recall on bait set on current iteration, and R_{last} is used to store the recall on U_{best} of the previous iteration. In each iteration, the U is divided into two subsets U_1 and U_2 by k-means (K=2) on those features selected with Algorithm 1. We then use R_{bait} to store the best recall between U_1 and U_2 (step 4), and the corresponding subset with best recall is also set to U_{best} (step 5). Then, the variations of the cluster size and the recall on bait set can be calculated by steps 6 and 7. When the size of suspicious cluster descends less than $\Delta S_{\min}$, or R_{bait} descends more than $\Delta R_{\max}$, the clustering process is terminated. The cluster after the last iteration is finally output as the player set of the specified categorization. These two termination conditions in step 2 can guarantee the precision and completeness of the extraction. While the suspicious cluster is too cohesive to be split, its size descends no more than $\Delta S_{\min}$, and then the iteration process is stopped promptly in case of further useless splitting. $\Delta R_{\max}$ is employed to prevent R_{bait} missing too much.

Fig. 3 gives a sample to demonstrate a typical process of the BBC. For visualization, the dimensions of raw data are

reduced to two by PCA. The bisected clusters are marked in purple and blue, and the known bait players denoted as $\times$. The results show that the data in red square in Fig. 3(a), where bait players gather, are iteratively refined from all the data. It can be easily found that the bait players concentrate almost at the upper left in Fig. 3(a)–(c). Until the 4th round in Fig. 3(d), bait players are scattered. Those unconcerned instances, which are far away from the bait players, are gradually removed due to lower R_{bait} of clusters they belong to. This iteration process will stop at the 4th round, because the bait players are partitioned, which causes $\Delta R > \Delta R_{\max}$ and meets the terminating conditions as defined in Algorithm 2. Then, we return the target group G from the third iteration, as the purple cluster in Fig. 3(c) shows, including the unlabeled players induced by those labeled bait players.

D. Hierarchical Parallelism on Varied Granularity

To categorize these players from billions of game log records in time, we need to employ the parallel processing framework of Apache Spark, which is a lightening-fast batch processing framework. An immutable distributed collection of data elements is represented as RDD in Spark, and RDD only enables row-level parallel computation, named as vertical parallelism. In our approach, we will also introduce the column-level parallel computation in the steps (c), (d), (e) of Fig. 1. As the computations in those steps are columns independent and can be parallelized at column-level, we call it horizontal parallelism.

In our proposed parallel framework, we define three parallelism based on the different parallel granularity from row and column as shown in Fig. 4.

PaS is the row-level parallelism provided by Apache Spark. As the blue dashed block shown in Fig. 4, P1, p2, p3, p4 are partitions of a large data set, which are computed in parallel across the nodes of the cluster.

PaLev is the parallelism that splits the vectors vertically into parallel parts by level. As the pink dashed block shown in Fig. 4, each divided part contains the entire features of a certain level in the log data.

PaCol is the parallelism that splits the vector of a level into parallel features to provide column-level acceleration. As the red dashed block shown in Fig. 4, the collection of elements with one feature can be operated in parallel.

These parallelisms on various granularities provide high efficient computation for our player categorization. As shown in Fig. 4, the data can be viewed as a large matrix, the PaS+PaCol will parallelly process the data in two directions: 1) the vertical direction is provided by Spark and 2) the horizontal direction is provided by local Scala parallel collection. With only the parallelism of Spark, data can be processed in parallel by row partitioning while different columns are still serial. In our hierarchical parallelism, columns are split into multi-RDD to be loaded in Scala parallel collection, e.g., RDD-e1, RDD-e2 and RDD-e3. Vertically, the RDD is computed in rows parallelism by Spark. Horizontally, a batch of RDD representing for different columns can also be processed in parallel by Scala. Comparing to the original Spark,

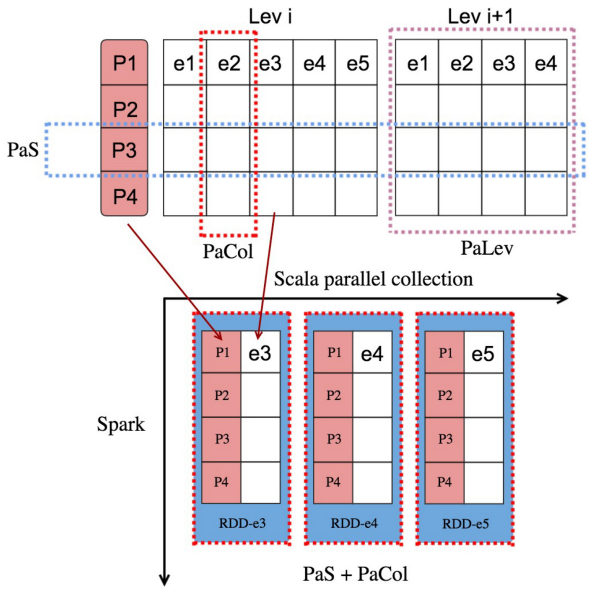


Fig. 4. HPF on varied granularity.

hierarchical parallelism achieves parallel computing in two dimensions.

As Fig. 1 shows, each step of feature representation needs parallelization technique to satisfy the big data processing. Steps (a) and (b) transform unstructured log data into player-level sequences by player aggregation, where only the parallelism Pas can be applied in these two steps. All the calculation on step (c) in Fig. 1 can be in parallel in the granularity of player levels, and different levels can be computed independently and, thus, we apply both the PaS and PaLev in this step. Similarly, we can also apply the PaS and PaLev in step (e), as it will execute the ranking-based feature selection based on the evaluation metrics defined on the granularity of player levels. While in step (d), the calculation, which is estimating the importance of each feature by clustering the quality of single feature, employing PaS and PaCol for parallel processing. Thus, our approach can flexibly provide a hierarchical parallelism instead of single strategy, each step of the process can achieve higher efficiency.

IV. Experiments

A. Experiments Setup

In our experiments, we employ the real log data of *New Qiannv Online*, a popular game produced by Netease. The data set is 468.5G, with 3 785 522 567 event records and 1 054 980 players, whose levels distributed from 1 to 40. In the categorization task, we employ two categories of the players in the following experiments: one is the mainline Bot, which is a prevalent harmful cheating script controlled by illegal studios to hunt game resources and trade for money, and the other is the RMB player who would rather pay for equipments and skills instead of devoting time to them. We manually refine the data set to find these two categories. There are 5104 mainline

TABLE I
CLUSTER HARDWARE AND SOFTWARES ENVIRONMENT

Hardware	Description	Software	Description
CPU	Intel Xeon E5-2630, $2 \times 6 \times 2$ cores per node	JDK	8u60
Memory	128G per node	Scala	2.10.5
Drive	2T per node	Spark	1.6.0
Physical Nodes	Huawei servers $\times 18$	Hadoop	2.6.0
Network	15.6Tbps	OS	CentOS 7.2

TABLE II
COMPUTATION RESOURCE USED IN FEATURE REPRESENTATION AND EXTRACTION ALGORITHM

	Driver	Executor	Partitions	Cores
feature representation	18G	6G $\times 80$	100	$6 \times$ Nodes
extraction algorithm	4G	1G $\times 20$	60	6×18

Bot players and 2691 RMB players for validation.[3] We execute all the experiments on our cluster environment, which is listed in Table I.

In the experiments, we employ yet another resource negotiator (YARN) as the resource framework to manage the cluster scale and resources, so that we can adjust the settings of YARN and Spark to achieve various resources allocation. The computation resources used in the feature representation and extraction algorithm in our evaluation experiments are given in Table II.

B. Determining the Optimal Size of Feature Subset

In the feature selection, the number of selected features N_{top} (subset size) in each level will obviously affects the complexity and accuracy of the extraction algorithm. The experiment in this section is to find the optimal N_{top} for the categorization algorithm. In order to compare the computation cost and categorization performance in the same scale, we normalize the computation speed into $[0, 1]$. The speed is commonly expressed by $V = \frac{1}{T}$, which is inversely proportional to the computation time T that the extraction algorithm runs. Then, the computation speed is scaled by

$$\text{SV} = \text{scaler} \cdot V, \text{scaler} = \frac{1}{\max(1/T)}. \tag{8}$$

Thus, we can plot the scaled speed and the $F1$ score of the output of Algorithm 2 for the specific category of players under various N_{top}, as shown in Fig. 5. As the selected feature number in each level goes up, $F1$-score presents an increasing trend in general, especially in the initial stages. When $N_{\text{top}} = 5$, the $F1$-score achieves the best value and stop increasing. Meanwhile, the speed keeps descending with the number of N_{top} increasing, and intersects with $F1$ curve at around $N_{\text{top}} = 5$. Then, we assign the subset size N_{top} as 5, to balance the accuracy and time cost of the extraction algorithm. In all the experiments of this article, the minimum size variation $\Delta S_{\min}$ and maximum recall descent $\Delta R_{\max}$ are all set as 0.1.

[3]These labeled players are all manually selected by some technical operation programmer in NetEase Company, thus can be regarded as the ground truths.

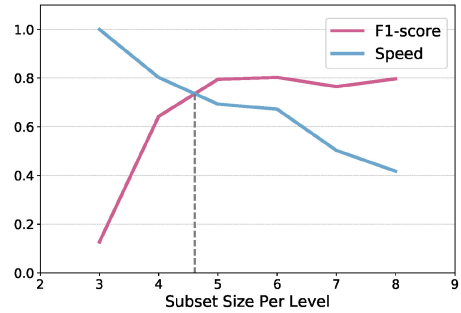


Fig. 5. Extraction algorithm speed and $F1$-score under varied selected feature size.

C. Categorization Performance Evaluation

The general approach for player categorization includes two steps: 1) feature representation and 2) extraction algorithm. Thus, we introduce different state-of-the-art feature representation methods and extraction algorithms, and then combine them in our evaluation experiments.

1) Feature Representation Methods and Semisupervised Extraction Methods: Besides our fixed-length action sequence (FAS) based feature representation method, there are other two feature representation methods: 1) unnormalized FAS (uFAS) and 2) game behavior characteristics features (BF) [1].

1) uFAS can be regarded as the simplified version of the FAS without normalizing the event frequencies among different levels. We introduce it to evaluate the importance of feature normalization in eliminating biases under varied levels.
2) BF [1] is designed based on the experiential behavior models of characteristics in MMORPGs, which are abstracted into battle, move, and collect categories, as a simplified and interpretable presentation.

To compare our BBC with other extraction methods, we employ two state-of-the-art semisupervised extracting methods: 1) LP and 2) two-step framework (TS). Among all these three methods, BBC, LP and TS, bait players play the same role in training as incomplete labels.

1) LP [15] labels the unknown samples based on their nearest labeled neighbors. Briefly speaking, if unlabeled sample A gets closer to labeled B than others, and then B is more likely to propagate its label to A. With the propagation going on, the decision boundaries will fall into lower density gaps, dividing the space into different classes. LP can be categorized into local and global consistency assumption, compared to the cluster assumption of BBC.
2) TS was first proposed in PU learning, which handles positive and unlabeled data set in document classification with two steps: a) filtering and b) refinement [32], [33]. Then, this framework is widely applied in mining specific groups, especially in anomaly detection [5], [34]–[36]. In our experiment, we refer to the TS method in [35], which applies unsupervised one-class SVM [37] for filtering

TABLE III
PERFORMANCE OF NINE METHODS FOR BOT PLAYER CATEGORY AND RMB PLAYER CATEGORY

Methods	Bot Player			RMB Gamer		
	R	P	$F1$	R	P	$F1$
FAS-BBC	0.9996 ± 0.0002	0.6793 ± 0.0202	0.8088 ± 0.0143	0.9088 ± 0.0416	0.3698 ± 0.0172	0.5251 ± 0.0167
uFAS-BBC	0.9791 ± 0.0042	0.6486 ± 0.0204	0.7801 ± 0.0140	0.9176 ± 0.0821	0.3038 ± 0.0456	0.4522 ± 0.0454
BF-BBC	0.9769 ± 0.0307	0.1321 ± 0.0034	0.2327 ± 0.0056	0.9070 ± 0.0410	0.2528 ± 0.0310	0.3943 ± 0.0404
FAS-LP	0.8496 ± 0.0159	0.7123 ± 0.0451	0.7739 ± 0.0268	0.7374 ± 0.0051	0.3319 ± 0.0066	0.4577 ± 0.0064
uFAS-LP	0.7467 ± 0.0087	0.7906 ± 0.0253	0.7679 ± 0.0141	0.8562 ± 0.0207	0.3295 ± 0.0114	0.4756 ± 0.0112
BF-LP	0.3312 ± 0.0265	0.5836 ± 0.0229	0.4215 ± 0.0196	0.3201 ± 0.0225	0.4277 ± 0.0221	0.3652 ± 0.0131
FAS-TS	0.2691 ± 0.0052	0.7310 ± 0.0126	0.3934 ± 0.0061	0.1779 ± 0.0089	0.3468 ± 0.0155	0.2350 ± 0.0098
uFAS-TS	0.1405 ± 0.0522	0.7247 ± 0.0773	0.2308 ± 0.0802	0.2564 ± 0.0178	0.9702 ± 0.0118	0.4052 ± 0.0216
BF-TS	0.2116 ± 0.0140	0.5184 ± 0.0282	0.3000 ± 0.0135	0.2747 ± 0.0878	0.2580 ± 0.0275	0.2629 ± 0.0508

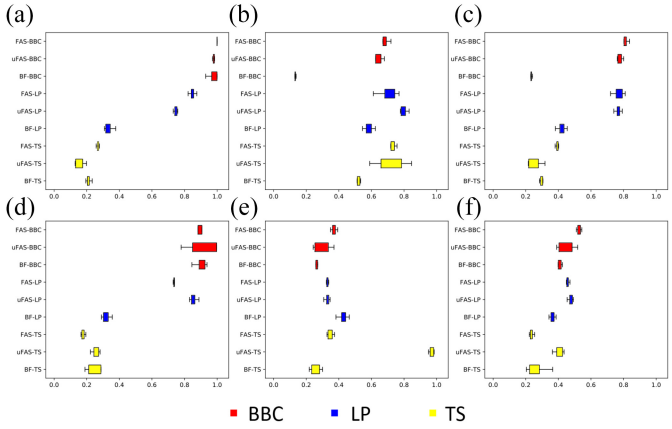


Fig. 6. Box plots of 3 × 3 methods comparison under 10-fold experiment. The location and width of each box indicate this method's performance and variance. (a) Recall for Bot Player. (b) Precision for Bot Player. (c) F1 for Bot Player. (d) Recall for RMB Player. (e) Precision for RMB Player. (f) F1 for RMB Player.

and local nonlinear SVM for refining. As it involves partial-informative labels, it can be also viewed as a semisupervised approach.

In the experiments of this section, these three feature representation methods: 1) FAS; 2) uFAS; and 3) BF, are combined pairwise with different extraction methods, BBC, LP, and TS. We then apply these nine methods to extract two specific player categories, respectively: 1) the Bot player and 2) RMB player. Considering the uncertainty of selecting bait players, each method is tested by 10 folds, and 10% players of the specific category are randomly chosen as the bait set in each fold. The recall, precision and $F1$ on the categories of Bot player and RMB player are calculated to evaluate each method's performance, and the calculation formula is similar to formula (4). The experimental results are given in Fig. 6 and Table III.

2) Comparing the Methods: Fig. 6(c) clearly suggests that FAS and uFAS outperforms BF on F1 evaluation. BF method abstracts the universal MMORPGs events into predefined feature set, which heavily depends on expert's experience. Then, features in BF method might ignore some helpful information beyond events, which is potential to be used in identifying players of specific category, while FAS and uFAS will keep most of such information, for example, the sequence information.

As Fig. 6(a) shows, BBC totally outperforms the other two methods in terms of recall. Although LP performs better in precision [Fig. 6(b)], it is still inferior to BBC slightly in $F1$ performance. This is consistent with the design target of BBC method that provides a solution to achieve good performances both on the precision and recall. These three methods with BBC have average recall above 0.97, as shown in Table III, while LP and TS are from 0.33 to 0.85 and from 0.21 to 0.27, respectively. It can be also observed that the worse recall performance of TS leads to the worse $F1$ performance, although it has good precision. The performance of the TS method relies heavily on the performance of the one-class SVM, which is used to extract the abnormal players. Basically speaking, the one-class SVM suits for detecting outliers, which are far from the normal ones, while the players of specific category may be distributed nearby the players of other categories as Fig. 3 shows. The results show that our proposed BBC is able to mine the target samples even if they are closed to others.

Fig. 6(d)–(f) shows similar results for the category of RMB players. FAS-BBC still achieves the best performance in $F1$ evaluation, and the methods whose extraction models are BBC can overall achieve the best recall performance. Comparing with Fig. 6(a) and (d), we observe that extraction models display similar pattern on categories of bot players and RMB players, BBC is better than LP, and LP is better than TS. On precision evaluation, the category of RMB players shows the lower performance than the category of Bot players for almost all the methods. The boxplot of Fig. 6(e) locates at the left of 0.5 except uFAS-TS. Due to the low precision, the category of RMB players has good recall performance but bad $F1$ performance, as shown in Fig. 6(d) and (f). Therefore, comparing with the categories of bot players and RMB players, the locations of boxplot in Fig. 6(a) are similar to those in Fig. 6(d), but those similarities are hardly shown between Fig. 6(c) and (f) due to the impact of precision. The reason can be inferred from different characteristics of two player categories. RMB players represent a subset of human players, while the code-driven Bot players whose behaviors are more

TABLE IV
PERFORMANCE OF VLSS, HRHC, AND FAS-BBC ON TWO DATASETS COLLECTED FROM DIFFERENT TIME INTERVALS

Methods	Dataset in May, 2018			Dataset in June, 2018		
	R	P	$F1$	R	P	$F1$
VLSS	0.8524± 0.0103	0.1959± 0.0224	0.3187± 0.0132	0.5321± 0.0172	0.6743± 0.0113	0.5947± 0.0126
HRHC [5]	0.5632±\	0.6843 ±\	0.6178±\	0.4371±\	0.7218±\	0.5444±\
FAS-BBC	0.7574± 0.0008	0.7677± 0.0196	0.7625± 0.0121	0.8002± 0.0163	0.9264± 0.0068	0.8587± 0.0105

[5] HRHC is an unsupervised method exploiting hierarchical clustering [40], Which is affected by the distance measure between samples and the linkage method between clusters. In the experiment, the distance measure, the linkage method, and the optimal number of the clusters for HRHC are determined and keeping unchanged, then there are no variances for HRHC.

different from human players. This result proves that RMB players are more cohesive with other human players than Bot players, making more challenge to identify the target group of RMB players precisely.

D. Further Experiment

As the behaviors of the Bot players will evolve quickly to avoid to be detected, the traditional Bot detection methods may be easy to fail. To further evaluate our method in that case, we employ additional two player log datasets collected from the same game, those new datasets consist of similar magnitude of event records and players with the datasets used in Section IV-A1, and each dataset contains the log data of one month, collected in May, 2018 and June, 2018. We manually labeled the Bot players, the number of validated Bot players is 1205 in the first dataset and 1498 in the second dataset. We also compare our approach with two Bot detection methods: the first one is modeling varying-length sequential data with supervised learning (VLSS) and the second one is a classical framework that uses histogram-based representation [3] and proceeds the histogram feature with hierarchical clustering (HRHC).

1) *VLSS:* We introduce the long short-term memory (LSTM) [38] to model the varying-length event log data. For the sake of fairness, we assume that the supervised VLSS contains the same prior knowledge as our FAS-BBC, thus we randomly choose 10% of Bot players and 10% normal players to construct the training set, and the remained data are used as test set for each dataset[4] in the following experiment.
2) *HRHC:* The first step of HRHC is transferring the sequences logs of the player into a histogram vector based on the action frequency of each player [3], and then we use the hierarchical clustering to group those players via their corresponding histogram vectors. In hierarchical clustering, the euclidean distance is used as the distance metric, the average-linkage distance is used as the distance between clusters, and the optimal number of clusters is chosen from a search range via the silhouette coefficient [39]. Finally, the group with the highest F1 score will be output.

For FAS-BBC, we use the same parameters as in the previous experiment. We evaluate precision, recall and F1 score on VLSS, HRHC, and our approach, each approach is executed ten times and the average evaluation is computed. The experimental results are shown in Table IV. As the Table IV shows, FAS-BBC outperforms VLSS on precision and F1 score in both two datasets. For the dataset collected in May, 2018, the recall of VLSS is better than FAS-BBC, while the precision of VLSS is much lower than FAS-BBC. Similar to other supervised methods, VLSS will heavily depend on the number of the training samples. Compared to HRHC, FAS-BBC is better on the three metrics in the two datasets. Bisecting clustering in FAS-BBC exploits the useful information of the labeled samples to ensure better performance of the target cluster, while the hierarchical clustering in HRHC is iteratively merging the nearest two clusters without other conducting information. In real application, the amount of labeled samples cannot always be enough, causing the supervised methods easily failed, while totally unsupervised methods are often unable to target at what we want precisely. Therefore, our framework is more efficient in practice.

[4] In practices, the newly evolved bot players are quite difficult to be detected, thus we usually can only confirm less 10% of those newly category of Bot players that can be used to train.

E. Parallelism Experimental Analysis

Our approach is implemented with the hierarchical parallel framework, utilizing techniques of Spark RDD and Scala parallel collection simultaneously. To verify the effectiveness of our framework, we remove parallelism of Palev and Pacol, meaning that only PaS provides row-based parallelism. This method is denoted as Spark only (SO) in the following experiment. The speed SV is calculated with (8). Moreover, we compute the speedup ratio SR [25] to reflect the comparable performance of our HPF over SO

$$\mathrm{SR}_{(\mathrm{HPF,SO})} = \frac{\mathrm{SV}_{\mathrm{HPF}}}{\mathrm{SV}_{\mathrm{SO}}}. \tag{9}$$

We compare HPF and SO under incremental cluster scale M_{nodes}, and the experimental results are given in Fig. 7. When the node number increases from 10 to 18, it presents a performance improvement for both HPF and SO, which indicates that the parallel framework will improve the computation efficiency. For all the spark nodes, our HPF always performs better than the SO framework, which suggests that our hierarchical parallelism is superior to the SO parallelism in computation efficiency.

When M_{nodes} is less than 18, HPF can grow more quickly than SO with the node number increasing, which can be also proved by the speedup ratio curve in Fig. 7. While in the case of $M_{\mathrm{nodes}} = 18$, the speedup ratio curve has a notable drop.

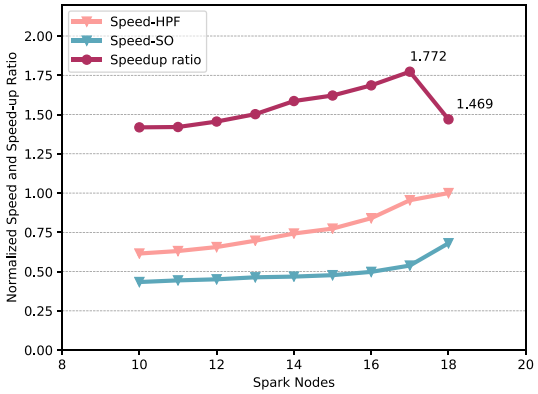


Fig. 7. Normalized speed of SO and HPF, and the speedup ratio $SR_{(HPF,SO)}$.

The reason may be that the temporal cost of the data exchange among multiple nodes has become larger than the temporal benefit from the increase of additional nodes when the node number is 18 for the same data scale. We call this condition as the critical point of the parallelism. As our HPF has employed both PaLev and PaCol for processing huge amounts of data, it will generate much more RDDs than the SO, and our HPF will reach the critical point faster than the SO.

V. Conclusion

This article proposed a novel framework on parallel categorizing MMORPGs players from real-world massive log data. In our framework, we employed the bait players, which is a small portion of the label-known target players, to extract the whole category of specific players from all the players. We proposed a structured feature representation of the player action sequences. With this novel feature representation method, we can acquire aligned, normalized, and informative features in lower dimensional space. Those aligned features are suitable to be fed to our proposed semisupervised extraction algorithm, which is based on the bisecting k-means clustering. We reduce the searching space iteratively in the extraction algorithm, using the recall ratio of bait players to guide the clustering process. Compared with the unsupervised methods, our semisupervised extraction approach exploits the incomplete labels to obtain the clustering more efficiently and effectively. Compared with binary classification, our approach can spend less temporal computation in highly customized feature representation and also can avoid the imbalanced-classes problem. To accelerate the whole categorization with more flexible parallelism, we proposed an HPF based on Apache Spark and Scala parallel collection. Our approach was evaluated by extensive experiments of the extraction for two different categories of players, and the results prove that our method outperforms other state-of-the-art methods. The results also proved the effectiveness of our HPF. As future work, we will extend our investigation to incorporate temporal feature of action sequence patterns into feature representation. Furthermore, social interactions of human players and Bot players may have significant differences, we will also explore the features in social network of game players to detect the suspicious category of players.

References

[1] Y. Chung *et al.*, "Detecting and monitoring game bots based on large-scale user-behavior log data analysis in multiplayer online games," *J. Supercomput.*, vol. 72, no. 9, pp. 3572–3587, 2016.
[2] A. R. Kang, J. Woo, J. Park, and H. K. Kim, "Online game bot detection based on party-play log analysis," *Comput. Math. Appl.*, vol. 65, no. 9, pp. 1384–1395, 2013.
[3] R. Thawonmas, Y. Kashifuji, and K. T. Chen, "Detection of MMORPG bots based on behavior analysis," in *Proc. Int. Conf. Adv. Comput. Entertainment Technol.*, 2008, pp. 91–94.
[4] M. van Kesteren, J. Langevoort, and F. Grootjen, "A step in the right direction: Botdetection in MMORPGs using movement analysis," in *Proc. BNAIC*, 2009, pp. 129–136.
[5] Y. Chung *et al.*, "Game bot detection approach based on behavior analysis and consideration of various play styles," *ETRI J.*, vol. 35, no. 6, pp. 1058–1067, 2013.
[6] C. Platzer, "Sequence-based bot detection in massive multiplayer online games," in *Proc. 8th Int. Conf. Inf. Commun. Signal Process. (ICICS)*, 2011, pp. 1–5.
[7] K. J. Shim, R. Sharan, and J. Srivastava, "Player performance prediction in massively multiplayer online role-playing games (MMORPGs)," in *Pacific-Asia Conf. Knowl. Discovery Data Mining*, pp. 71–80. Heidelberg, Germany: Springer, 2010.
[8] K. J. Shim and J. Srivastava, "Behavioral profiles of character types in everquest II," in *Proc. IEEE Symp. Comput. Intell. Games (CIG)*, 2010, pp. 186–194.
[9] M. A. Ahmad, B. Keegan, J. Srivastava, D. Williams, and N. Contractor, "Mining for gold farmers: Automatic detection of deviant players in mmogs," in *Proc. Int. Conf. Comput. Sci. Eng.*, vol. 4, 2009, pp. 340–345.
[10] C. Bauckhage, A. Drachen, and R. Sifa, "Clustering game behavior data," *IEEE Trans. Comput. Intell. AI Games*, vol. 7, no. 3, pp. 266–278, Sep. 2015.
[11] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau, "Guns, swords and data: Clustering of player behavior in computer games in the wild," in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, 2012, pp. 163–170.
[12] C. Thurau and C. Bauckhage, "Analyzing the evolution of social groups in world of warcraft®," in *Proc. IEEE Symp. Comput. Intell. Games (CIG)*, 2010, pp. 170–177.
[13] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2009.
[14] S. Bandyopadhyay, G. Chowdhary, and D. Sengupta, "FOCS: Fast overlapped community search," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 2974–2985, Nov. 2015.
[15] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Sch. Comput., Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CALD-02-107, 2002.
[16] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Proc. NIPS*, vol. 16, 2003, pp. 321–328.
[17] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, 2007, Art. no. 036106.
[18] J. Xie and B. K. Szymanski, "Community detection using a neighborhood strength driven label propagation algorithm," in *Proc. Netw. Sci. Workshop (NSW)*, 2011, pp. 188–195.
[19] F. Wang and C. Zhang, "Label propagation through linear neighborhoods," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 1, pp. 55–67, Jan. 2008.
[20] Z. Zhang, M. Zhao, and T. W. S. Chow, "Graph based constrained semi-supervised learning framework via label propagation over adaptive neighborhood," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 9, pp. 2362–2376, Sep. 2015.
[21] J. Dass, V. Sarin, and R. N. Mahapatra, "Fast and communication-efficient algorithm for distributed support vector machine training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1065–1076, May 2019.
[22] F. Xu, H. Zheng, H. Jiang, W. Shao, H. Liu, and Z. Zhou, "Cost-effective cloud server provisioning for predictable performance of big data analytics," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1036–1051, May 2019.
[23] N. Bharill, A. Tiwari, and A. Malviya, "Fuzzy based scalable clustering algorithms for handling big data using apache spark," *IEEE Trans. Big Data*, vol. 2, no. 4, pp. 339–352, Dec. 2016.
[24] D. Savage, X. Zhang, P. Chou, X. Yu, and Q. Wang, "Distributed mining of contrast patterns," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 7, pp. 1881–1890, Jul. 2017.

[25] J. Chen *et al.*, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 919–933, Apr. 2017.
[26] M. Steinbach, G. Karypis, and V. Kumar, "A comparison of document clustering techniques," in *Proc. KDD Workshop Text Min.*, vol. 400. Boston, MA, USA, 2000, pp. 525–526.
[27] Q. Song, J. Ni, and G. Wang, "A fast clustering-based feature subset selection algorithm for high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 1, pp. 1–14, 2013.
[28] D. L. Donoho and J. Tanner, "Neighborliness of randomly projected simplices in high dimensions," *Proc. Nat. Acad. Sci. United States America*, vol. 102, no. 27, pp. 9452–9457, 2005.
[29] J. Dai, Q. Hu, J. Zhang, H. Hu, and N. Zheng, "Attribute selection for partially labeled categorical data by rough set approach," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2460–2471, Sep. 2017.
[30] Y. Liu, F. Tang, and Z. Zeng, "Feature selection based on dependency margin," *IEEE Trans. Cybern.*, vol. 45, no. 6, pp. 1209–1221, Jun. 2015.
[31] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 5, pp. 603–619, May 2002.
[32] X. Li and B. Liu, "Learning to classify texts using positive and unlabeled data," in *Proc. Int. Joint Conf. Artif. Intell.*, 2003, pp. 587–592.
[33] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu, "Building text classifiers using positive and unlabeled examples," in *Proc. IEEE Int. Conf. Data Min.*, 2003, pp. 179–186.
[34] X. Yu, L. A. Tang, and J. Han, "Filtering and refinement: A two-stage approach for efficient and effective anomaly detection," in *Proc. 9th IEEE Int. Conf. Data Min.*, 2009, pp. 617–626.
[35] B. Du, C. Liu, W. Zhou, Z. Hou, and H. Xiong, "Catch me if you can: Detecting pickpocket suspects from large-scale transit records," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2016, pp. 87–96.
[36] S. Zhao *et al.*, "Discovering different kinds of smartphone users through their application usage behaviors," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2016, pp. 498–509.
[37] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, and J. C. Platt, "Support vector method for novelty detection," in *Proc. NIPS*, vol. 12, 1999, pp. 582–588.
[38] J. Tao, J. Xu, L. Gong, Y. Li, C. Fan, and Z. Zhao, "Nguard: A game bot detection framework for NetEase MMORPGs," in *Proc. 24nd ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2018. pp. 811–820.
[39] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *J. Comput. Appl. Math.*, vol. 20, pp. 53–65, Nov. 1987.
[40] F. Corpet, "Multiple sequence alignment with hierarchical clustering," *Nucleic Acids Res.*, vol. 16, no. 22, pp. 10881–10890, 1988.

Jing Fan received the B.S. degree in computer science and engineering from Shandong University, Jinan, China, in 2007, and the M.S. degree in software engineering from Zhejiang Normal University, Jinhua, China, in 2011. She is currently pursuing the Ph.D. degree with the College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

Her current research interests include machine learning, big data mining, and granular computing.

Shaowen Gao received the B.S. degree in automation from EE College, Zhejiang University, Hangzhou, China, in 2015, where he is currently pursuing the M.S. degree with the College of Control Science and Engineering.

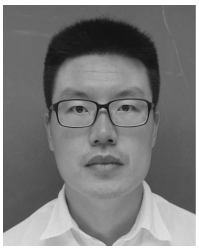
His research interests include mass data mining and parallel computing.

Yong Liu (Member, IEEE) received the B.S. degree in computer science and engineering and the Ph.D. degree in computer science from Zhejiang University, Zhejiang, China, in 2001 and 2007, respectively.

He is currently a Professor with the Institute of Cyber-Systems and Control, College of Control Science and Engineering, Zhejiang University. He has published over 60 research papers in machine learning, computer vision, information fusion, and robotics. His current research interests include machine learning, robotics vision, and information fusion.

Xinqiang Ma received the B.S. degree in computer science and technology from Yantai University, Yantai, China, in 2004, and the M.S. degree in computer application and technology from Guizhou University, Guiyang, China, in 2007.

He is currently a Professor withăthe the Institute of Intelligent Computing and Visualization based on Big Data,ăChongqing University of Arts and Sciences, Chongqing, China. His current research interests include machine learning, data mining, and visual analytics.



Jiandang Yang received the B.S. degree in computer science and technology from Shanghai University of Electric Power, Shanghai, China, in 2008, and the M.S. degree in computer application and technology from Hangzhou Dianzi University, Hangzhou, China, in 2013.

He is currently an Assistant Research Fellow with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou. His current research interests include machine learning, big data mining, and visual analytics.

Changjie Fan received the B.S. degree in automation from the Department of Automation, University of Science and Technology of China, Hefei, China, in 2003, and the Ph.D. degree in computer science from the University of Science and Technology of China in 2007.

He is currently an employee of Netease, Hangzhou, China, where he is currently the Director of the Game AI Lab. His research interests include machine learning, natural language processing, and reinforcement learning.