When cyberathletes conceal their game: Clustering confusion matrices to identify avatar aliases

Olivier Cavadenti, Victor Codocedo, Jean-François Boulicaut, Mehdi Kaytoue

When Cyberathletes Conceal Their Game: Clustering Confusion Matrices to Identify Avatar Aliases ⋆

Olivier Cavadenti, Victor Codocedo, Jean-François Boulicaut and Mehdi Kaytoue
Université de Lyon. CNRS, INSA-Lyon, LIRIS. UMR5205, F-69621, France.
contact:*firstname.name@insa-lyon.fr*

***Abstract*—Video game is a very lucrative industry, unleashed by the ubiquity of gaming devices, multi-player networks and live broadcasting platforms. Games generate large amounts of behavioural data which are valuable to face the new challenges of *video game analytics* such as detecting balance issues, bugs and cheaters. In electronic sports (e-sports), cyberathletes conceal their online training using different aliases or avatars (virtual identities), which allow them not being recognized by the opponents they may face in future competitions (with cash prices challenging already most of the traditional sports). It was recently suggested that behavioural data generated by the games allows predicting the avatar associated to a game play with high accuracy. However, when a player uses several avatars, accuracy drastically drops as prediction models cannot easily differentiate the player's different avatar aliases. Since mappings between players and avatars do not exist, we introduce the *avatar aliases identification problem* and propose an original approach for alias resolution based on supervised classification and Formal Concept Analysis. We thoroughly evaluate our method with the video game *Starcraft 2* which has a very wide and active community with players from diverse cultures and nations. We show that under some circumstances, the avatars of a given player can easily be recognized as such. These results are valuable for e-sport structures (to help preparing tournaments), and game editors (detecting cheaters or usurpers).**

I. Introduction

Currently, video games are a popular and lucrative industry generating more revenue than both Cinema and Music. This recent and fast development has been catalysed by several factors. First, an increasing part of the population has access to connected electronic devices (computers, smart-phones, etc.), and to the Web including a myriad of single and/or multi-player games. Video games are now a popular leisure activity. Furthermore, the recent development of competitive gaming and live video game streaming platforms [8] makes electronic sports (e-sports) a reality [15]. Professional players on contract with teams and sponsors are taking part in competitions with cash prices challenging already most of the traditional sports. Cyberathletes are widely followed in so-called *off-line events* and daily supported on the Web through their own live broadcasts [8] generating the fourth largest stream of data in 2015 for the platform, *Twitch.tv*. People not only enjoy playing, but also enjoy watching the others for a variety reasons [3], [8]. Games generate tremendous amounts of behavioural data, valuable to face the many new industrial challenges such as detecting balance issues [2], bugs and cheaters [18], but also valuable for several academic fields such as artificial intelligence [12], computer human interactions [21] and cognitive sciences [16].

In this paper, we introduce a new problem: the *avatar aliases identification problem*. In most of online competitive games, players need an "avatar" (i.e., an online identity) to log in the game network. Nothing forbids a player to have several avatars and actually, it is a very common practice for professional players, or cyberathletes, of several video game franchises (e.g. *League of Legends*, *Starcraft 2*, *Dota 2*, *Counter Strike*). Players generally have one official avatar for official tournaments, and several others to conceal their tactics while playing without being recognized by other players they may meet online: global rankings and leagues are public just as in chess and tennis, while game logs (called replays) are available and prone to analysis by means of visualization[1], machine learning and data mining techniques [9], [14] just as in standard sport analytics. Accordingly, we are facing a set of players, generating behavioural data (game logs or traces), in a one-to-many relationship with avatars (the many-to-many relationship will be discussed later). Our goal is not to discover this mapping, since players privacy is protected. Instead, the *avatar aliases identification problem* aims at discovering groups of avatars belonging to the same player. Solving this problem is motivated by the growing need of e-sport structures to study the games and strategies of the opponents (tournament preparation), and the security challenges of game editors (detecting usurpers).

Recently, it was suggested that behavioural data hide individual identifying patterns [16], [21], (tested on the real time strategy game *Starcraft 2* (©Blizzard)). After choosing features related to keyboard usage, Yan et al. showed that a classifier can be trained to predict with high accuracy the avatars involved in a game play [21]. Nevertheless, they purposely considered datasets without players having several avatars (what we call avatar aliases). Indeed, in presence of such *avatar aliases*, the prediction accuracy drastically degrades, since prediction models fail at differentiating two avatars of the same player. We build on this work by proposing an approach called DÉBROUILLE for answering the *avatar aliases identification problem*: it relies on mining confusion matrices yielded by a supervised classifier using Formal Concept Analysis [5], and exploits the confusion a classifier has in presence of avatar aliases when they belong to the same player. Experimental evaluation shows very good results under certain restrictions imposed to the set of players involved in a dataset. The remainder of this paper is organized as

⋆ This research has been partially funded by the French National Project FUI AAP 14 Tracaverre 2012-2016.

[1]For example, https://sites.google.com/site/sc2gears/ for *Starcraft 2*

follows. Section II introduces the problem settings, our claims and goals. We formally present an original methodology in Section III for discovering avatar aliases and the evaluation strategy in Section IV. Our results are assessed through an extensive evaluation in Section V. Related work is discussed in Section VI before we conclude.

II. Problem Settings

A. Starcraft 2 in competitive gaming

We study the *real-time strategy* game (RTS) *Starcraft 2*, a franchise released in 2010 which met success in competitive gaming and e-sports [15]. A standard game involves two players and each chooses a faction (Zerg, Protoss or Terran) with different weaknesses and strengths (following a rock/paper/scissors principle). Players control buildings and units through an aerial view of the battleground map, collect resources to build an army and achieve victory by destroying the opponent's forces. A *player* needs an *avatar* to enter the dedicated multi-player system called `Battle.net`. In the context of e-sport, our careful attention is given to the *avatar aliases identification problem* that affects the game[2]. Nothing forbids a user from having multiple avatars, which is actually a common practice among professional players for a variety of reasons. For example, as they compete in international tournaments, they may need an avatar in each different *Battle.net* server (USA, Europe, Korea, etc.). Another important reason is that professional users may need to hide their tactics while training before competitions. For this reason, cyberathletes may create different avatars with names such as `lIlIlIlIlIlll` called *bar code avatars* (as in Table III) for practising on public servers without being recognized. Practise on public networks corresponds to a major time of their activity [15]. Currently, among the best 50 avatars in the global *Starcraft 2* ranking system, 46 avatars are *bar code*[3]. As argued in the introduction, multi-aliases prompt several problems. Our goal is to formalize and solve the *avatar aliases identification problem* which consists in finding avatars belonging to the same player without any information about the individual but his game behaviour.

B. Behavioural data

During a game, a *player* performs a series of *actions* which generates a *trace*. Basic actions rely on mouse usage, provided with semantics: "selection of a unit", "attack with selected unit", "collect resources with selected unit", etc. Many (if not all) of these actions can be performed with the mouse by selecting a series of menus on the screen. However, to improve the amount of actions a user can perform per minute (a critical issue in this kind of video games), expert players prefer to bind these actions directly to the keyboard through the use of customized *hotkeys*, also called *control groups* [21]. As such, any action, done through mouse or keyboard, is stored in *replay files* that are available on the Web and usable for observing and studying other players strategies. Replays constitute extremely rich behavioural data, and we use them for answering our problem. Table I shows two traces associated to a single game: Player *TAiLS* starts by selecting with the mouse a building called *Base*, assigns it to hotkey 1 (a), selects some units with the mouse (s), select units attached to hotkey 2 (s), etc.

Avatar	Game trace	Outcome
`RorO`	s,s,hotkey4a,s,hotkey3a,s,hotkey3s, ...	Lose
`TAiLS`	Base,hotkey1a,s,hotkey1s,s,hotkey1s, ...	Win

Table I: Traces of a match for two players

C. Predicting avatars from behavioural traces

Let us consider the following scenario. Given a trace, is it possible to find its associated avatar? In Table I, given the trace in the first row, is it possible to say that the avatar associated to it is `RorO`? We can approach this question from a classification point of view. Given a training set of traces labelled with their associated avatars, we would like to classify new "instances" (new traces) in the space of classes represented by the set of avatars. As such, we can apply any supervised classification technique in this direction. Recently, Yan et al. [21] suggested that behavioural data, as presented in the previous subsection, offer predictions with high accuracy, when there are no aliases in the dataset. In presence of aliases however, the accuracy drastically drops, as trained models are not able to differentiate properly different avatars of a same individual.

The reason why game traces allow good avatar predictions, when there are no aliases, is related to the cognitive process involved in the activity of playing [16]. A game of *Starcraft 2* is short (between 15 and 20 minutes in average), in real time, and several tasks and strategies are repeated thousands of time to be achieved instinctively: mastering the game requires an intense multi-tasking activity (not sequential) and daily practise. Novice's traces will seem messy during the learning process, but will converge with practise towards distinctive patterns, or individual *signatures*.

D. Towards confusion clustering to identify avatars aliases

In general, players use several avatars that generate traces. This is modelled in Figure 1. The mapping $owns : Player \rightarrow Avatar$ is however not known (or very partially by the game editor that keeps it private). The only information available is the mapping $generates : Avatar \rightarrow Trace$. Notice that we make the assumption that an avatar belongs to a single player (individual). This may not hold: nothing forbids two persons from sharing an avatar. However, as an avatar is ranked according to *all* its games (wins and loses) in a world ranking system, it is fairly safe to assume in general that professional players (our focus) do not share their avatars to protect their *reputations* but also *privacy* when playing in secrecy.

The general intuition of our approach to tackle the *avatar aliases identification problem* is now introduced. Our model can be split in two sub-tasks, namely "finding Trace patterns associated to Avatars" and "finding Avatar clusters associated to Players". For the first task, a classifier is built as explained in the previous subsection, for predicting the avatar involved in a game. In presence of avatar aliases, its accuracy drastically drops. However, it is a fair hypothesis that the classifier confusion shall be spread locally among avatar aliases. As such, the second step involves a particular clustering of the confusion matrix, that outputs avatar aliases candidates. We formalize this step with FCA: a fuzzy set operator on class memberships enables the finding of candidate patterns.

[2]Multi-aliases are known as "smurfs" players within the gamer community.
[3]http://www.sc2ranks.com visited on 2015, May 28^{th}

Figure 1: A simple model of game traces generations

III. Method Formalization

Let A be a set of avatars and T be a set of traces such that for a given avatar $a \in A$, the set $T_a \subseteq T$ is the set of all traces generated by a. Consider a classifier ρ where labels are the avatars to predict. Our method consists in using the confusion matrix generated by a classifier ρ and analyse how confusion between labels is spread to extract candidates of avatar aliases. This process has two steps: firstly, specific patterns are extracted from the confusion matrix, secondly, they are scored, ranked and post processed. The corresponding pseudo-code of our approach, called DÉBROUILLE for *unscramble* in French, is given in Algorithm 1 and is detailed hereafter.

A. Classifying Traces

A classifier is a function $\rho : T \to A$ that assigns the avatar $\rho(t) \in A$ to a given trace $t \in T$. Let $n = |A|$ be the number of avatars in A, from any classifier ρ, one can derive a confusion matrix

$$C^{\rho}_{n \times n} = (c_{i,j})$$

where

$$c_{i,j} = |\{t \in T_{a_i} \ s.t. \ \rho(t) = a_j\}|$$

Each row and column of C^{ρ} correspond to an avatar, while the value c_{ij} is the number of traces of avatar a_i that are classified by ρ as of avatar a_j. The normalized confusion matrix is given by

$$\tilde{C}^{\rho} = [c_{i,j}/|T_{a_i}|]$$

where $\tilde{C}^{\rho}_{i,i} = 1$ for any $i \in [1, |A|]$ means all the traces of avatar a_i are correctly classified by ρ.

Table II gives an example of normalized confusion matrix between five avatars. It is worth mentioning that we have generalized the classifier to a given function ρ and thus, we will treat it as a "black box" with an input (in the form of a set of traces T and avatars A) and an output which corresponds to the confusion matrix. Internally, features built from traces can be chosen independently, and the classifier can split the set of traces into train and test sets using different strategies.

Algorithm 1 DÉBROUILLE pseudo code

Require: $\tilde{C}^{\rho}$: normalized confusion matrix, λ cluster score threshold, s score threshold.
Ensure: P list of pairs of avatar ranked by cluster score.

```
P ← ∅
C ← MineFuzzyConcepts(C̃^ρ, s)
C ← rank C according to s
for all c ∈ C do
    pairs ← pairs from the pattern concept extent of c
    for all p ∈ pairs do
        if p ∉ P and cluster_score(p) > λ then
            P ← P ∪ {p}
P ← rank P according to the cluster score
return P
```

B. Clustering Avatars

Our goal is to discover groups of avatars that belong to the same player. Our intuition is that a classifier will hardly differentiate these avatar aliases, hence the confusion matrix values should be high and concentrated around them. This is exemplified in Table II: avatars $\{a_1, a_2\}$ are candidates to belong to the same player, $\{a_4, a_5\}$ shall belong to another user, while a_3 stays as singleton with a diagonal high value. Hence, a reasonable clustering of avatars would be given by the partition $\{\{a_1, a_2\}, \{a_3\}, \{a_4, a_5\}\}$.

	a_1	a_2	a_3	a_4	a_5
a_1	0.6	0.4	0	0	0
a_2	0.4	0.55	0.05	0	0
a_3	0	0	0.8	0.15	0.05
a_4	0	0.05	0	0.7	0.25
a_5	0	0	0	0.5	0.5

Table II: Confusion Matrix example

More formally, given a normalized confusion matrix $\tilde{C}^{\rho}$, we would like to find pairs of avatars $a_i, a_j \in U$ such that $\tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{ji} \simeq \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{jj}$ and $\tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ji} + \tilde{C}^{\rho}_{ii} + \tilde{C}^{\rho}_{jj} \simeq 2$. These conditions come from the fact that, if a_i, a_j correspond to the same player, traces in T_{a_i} have the same probability of being classified as a_i or a_j (the same for traces in T_{a_j}). Furthermore, for a trace of avatar a_i, it is required that the probability of classification is spread between a_i and a_j only, meaning that $\tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ii} \simeq 1$ (similarly for a_j).

Using this rationale, in what follows we propose (i) to extract patterns from the confusion matrix, and (ii) to post process them to provide groups of candidate avatar pairs. The first step is achieved thanks to Formal Concept analysis (FCA [5], [6]), while we define scoring functions and ranking for the second step.

1) Fuzzy pattern structure: Let us define the fuzzy set of membership degrees L^A where $L = [0, 1]$, such as the mapping function $\delta : A \to L^A$ assigns membership values for the avatar a_i in the fuzzy set L^A based on the normalized confusion matrix. Simply, this is a mapping that assigns to a_i its corresponding row in $\tilde{C}^{\rho}$ which we denote $\tilde{C}^{\rho}_i$.

We model accordingly a confusion matrix $\tilde{C}^{\rho}$ as a pattern structure $(A, (L^A, \sqcap), \delta)$. The operator $\sqcap$ is a meet operator in a semi-lattice (idempotent, commutative and associative), and is defined as follows, given two avatars $a_i, a_j \in A$:

$$\delta(a_i) \sqcap \delta(a_j) = \langle min(\tilde{C}^{\rho}_{ik}, \tilde{C}^{\rho}_{jk}) \rangle, \ k \in [1, |A|]$$
$$\delta(a_i) \sqsubseteq \delta(a_j) \iff \delta(a_i) \sqcap \delta(a_j) = \delta(a_i)$$

Actually, $\sqcap$ corresponds to the fuzzy set intersection and $(L^A, \sqsubseteq)$ is a partial order over the elements of L^A which can be represented as a semi-lattice.

The pattern structure $(A, (L^A, \sqcap), \delta)$ is provided with two derivation operators, forming a Galois connection [6]. Formally, we have, for a subset of avatars $\mathtt{A} \subseteq A$ and a fuzzy set $\mathtt{d} \in L^A$ such as:

$$\mathtt{A}^{\square} = \prod_{a \in \mathtt{A}} \delta(a) \qquad \mathtt{d}^{\square} = \{a \in A \mid \mathtt{d} \sqsubseteq \delta(a)\}$$

A pair $(\mathtt{A}, \mathtt{d})$ is a pattern concept iff $\mathtt{A}^{\square} = \mathtt{d}$ and $\mathtt{d}^{\square} = \mathtt{A}$. Pattern concepts are ordered by extent inclusion such that for $(\mathtt{A}_1, \mathtt{d}_1)$ and $(\mathtt{A}_2, \mathtt{d}_2)$ we have:

$$(\mathtt{A}_1, \mathtt{d}_1) \leq (\mathtt{A}_2, \mathtt{d}_2) \iff \mathtt{A}_1 \subseteq \mathtt{A}_2 \,(\text{or } \mathtt{d}_1 \sqsupseteq \mathtt{d}_2)$$

Intuitively, a pattern concept $(\mathtt{A}, \mathtt{d})$ contains a fuzzy set $\mathtt{d}$ which can be represented as a *vector* $\mathtt{d} = \langle \mathtt{d}^j \rangle$ with length $|A|$ where each value $\mathtt{d}^j$ is the minimum for all rows i in column j of matrix $\tilde{C}^{\rho}$ such that $a_i \in \mathtt{A}$.

Example. The Table II illustrates a confusion matrix involving five avatars. It has been obtained from a classifier ρ. We illustrate first how pattern concepts are generated:

$$\begin{aligned} \delta(a_1) &= \{a_1^{0.6}, a_2^{0.4}, a_3^{0}, a_4^{0}, a_5^{0}\} \\ \delta(a_2) &= \{a_1^{0.4}, a_2^{0.55}, a_3^{0.05}, a_4^{0}, a_5^{0}\} \\ \delta(a_1) \sqcap \delta(a_2) &= \{a_1^{0.4}, a_2^{0.4}, a_3^{0}, a_4^{0}, a_5^{0}\} \end{aligned}$$

2) Scoring concepts and extracting aliases: The scoring function $s : L^A \to [0, 1]$ is given as follows: for a pattern $\mathtt{d}$,

$$s(\mathtt{d}) = \sum_{j=1}^{|A|} \mathtt{d}^j$$

It is clear that function s is decreasing w.r.t. the order of pattern concepts, i.e. $(\mathtt{A}_1, \mathtt{d}_1) \leq (\mathtt{A}_2, \mathtt{d}_2) \implies s(\mathtt{d}_1) \leq s(\mathtt{d}_2)$. Thus, pattern concepts can be mined up to a given score threshold analogously as formal concepts can be mined up to a given minimal support, as it is done in pattern mining [1]. We can appreciate that the higher the score of a given pattern, the more *confused* is the classification of traces of avatars $a \in \mathtt{A}$ by ρ in $\tilde{C}^{\rho}$ and thus, they become candidates for merging. This property directly follows from the choice of our similarity operator $\sqcap$ as a fuzzy set intersection, which behave as a pessimistic operator (returning minimum values).

The pattern mining step is executed as follows and corresponding to the $MineFuzzyConcepts$ step (Line 2) in Algorithm 1. From the confusion matrix we compute all possible pattern concepts using the `addIntent` algorithm [17]. Pattern concepts are then ranked according to their score (Line 3) and converted into a list of pairs (Line 5). For example, if a pattern concept extent contains three avatars a_1, a_2 and a_3, we convert this concept into pairs (a_1, a_2), (a_1, a_3) and (a_2, a_3). The order among pairs of the same pattern is disregarded.

The `addIntent` algorithm is known to have a linear complexity w.r.t. the number of possible pattern concepts [17] which can grow exponentially w.r.t. the number of avatars in A. In our case, experimental evidences suggest that the number of pattern concepts in this setting is much lower than A^2, given the empty intersection between most pairs of avatars in A. Finally, given that the scoring function is monotonous w.r.t. the order $\sqsubseteq$, it can be used as a filter to stop the calculation of meaningless patterns.

Example. Continuing the previous example, we have:

$$s(\{a_1, a_2\}^{\square}) = 0.8 \quad (1)$$

$$s(\{a_4, a_5\}^{\square}) = 0.75 \quad (2)$$

$$s(\{a_1, a_2, a_4\}^{\square}) = 0.05 \quad (3)$$

3) Post-processing candidates: Consider the clustering condition previously formalized as $\tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{ji} \simeq \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{jj}$ and $\tilde{C}^{\rho}_{ii} + \tilde{C}^{\rho}_{ij} + \tilde{C}^{\rho}_{ji} + \tilde{C}^{\rho}_{jj} \simeq 2$. Consider that the pair of avatars (a_i, a_j) respects these conditions. It is easy to see that (a_i, a_j) will necessarily be a candidate pair highly ranked from the previous step.

$$\begin{aligned} \tilde{C}^{\rho}_{ij} \simeq \tilde{C}^{\rho}_{jj} &\simeq min(\tilde{C}^{\rho}_{ij}, \tilde{C}^{\rho}_{jj}) \\ \tilde{C}^{\rho}_{ii} \simeq \tilde{C}^{\rho}_{ji} &\simeq min(\tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ji}) \\ \implies min(\tilde{C}^{\rho}_{ij}, \tilde{C}^{\rho}_{jj}) + min(\tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ji}) &\simeq 1 \end{aligned}$$

Thus, the set of avatar clusters we are looking for are contained within the set of candidate pairs and moreover, they are highly ranked. In order to filter the list of candidates from pairs that do not hold the avatar cluster definition, we propose a cosine similarity measure between a couple of vectors calculated for each avatar as follows. Let (a_i, a_j) be a candidate pair, the cluster score is defined as:

$$cluster\_score(a_i, a_j) = cosine(\langle \tilde{C}^{\rho}_{ii}, \tilde{C}^{\rho}_{ij} \rangle, \langle \tilde{C}^{\rho}_{jj}, \tilde{C}^{\rho}_{ji} \rangle)$$

The cluster score establishes a measure of how close is a candidate pair from being an avatar cluster. The logic of this follows from the following scenario. Consider that the traces of avatar a_i were all correctly classified meaning that $\tilde{C}^{\rho}_{ii} = 1$ and that the traces of avatar a_j were all incorrectly classified as a_i, meaning that $\tilde{C}^{\rho}_{ji} = 1$, thus we have the following section of the normalized confusion matrix:

	a_i	a_j
a_i	1	0
a_j	1	0

We can observe that the pair (a_i, a_j) will be contained in the set of candidate pairs and will be highly ranked, even though it is not an avatar cluster since it violates the first condition. The cluster score for this particular case can be calculated as:

$$cluster\_score(a_i, a_j) = cosine(\langle 1, 0 \rangle, \langle 0, 1 \rangle) = 0$$

meaning that this candidate pair is not an avatar cluster. Notice that for the pair of avatars such that $a_{ii} = 1$ and $a_{jj} = 1$, the cluster score is 1 (cosine between parallel vectors) while the pair is not an avatar cluster. Indeed, this is true, however this pair would have a score s equal to 0 and would be at the bottom of the ranked candidate pairs. A third kind of pair occurs when the traces of a_i and a_j are all incorrectly classified as a third avatar a_k. In such a case, the cluster score is 0.

The post processing step is executed as follows as depicted in Algorithm 1. Given a ranked list of candidate pairs yielded from the previous step (Line 2 and 3), each pair is evaluated using the cluster score. Given an arbitrary threshold λ, if the cluster score of the candidate pair is below this threshold, then it is rejected (Line 7 and 8). Candidate pairs are re-ranked into a final list of avatar clusters (Line 9).

IV. Evaluation

In this section, we provide a detailed evaluation procedure used in the experiments for assessing the ability of our approach at finding avatar aliases.

Account: (eu,2452136)	
Avatar	**URL**
MinChul	http://eu.battle.net/sc2/en/profile/2452136/1/MinChul/
SKMC	http://eu.battle.net/sc2/en/profile/2452136/1/SKMC/
Account: (eu,4233584)	
Avatar	**URL**
INnoVation	http://eu.battle.net/sc2/en/profile/4233584/1/INnoVation/
lIlIllIlIlll	http://eu.battle.net/sc2/en/profile/4233584/1/lIlIlIlIlIlll/
Account: (us,288081)	
Avatar	**URL**
Minigun	http://us.battle.net/sc2/en/profile/288081/1/Minigun/
ROOTMinigun	http://us.battle.net/sc2/en/profile/288081/1/ROOTMinigun/
Account: (us,2929052)	
Avatar	**URL**
ROOTheognis	http://us.battle.net/sc2/en/profile/2929052/1/ROOTheognis/
Account: (us,3023756)	
Avatar	**URL**
MinChul	http://us.battle.net/sc2/en/profile/3023756/1/MinChul/

Table III: An example of five Battle.net accounts and their respective avatars

A. Avatars matching

As we do not have information about the users behind the avatars, it is not possible to actually evaluate the candidate pairs for merging using a "ground truth". Instead, we perform an indirect evaluation of our approach using three different strategies one being specific to the game *Starcraft 2*. Indeed, as illustrated in Figure 2, the avatar system of *Starcraft 2* is more elaborated than our general model given in Figure 1. The Table III exemplifies this model.

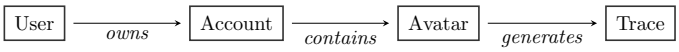

Figure 2: The trace generation model in *Starcraft 2*

1) Nicknames: Each avatar is associated with a non-unique (nick-)name. Names are chosen by users and **can** be changed at any time. To change the name, users have to pay a fee to the videogame company. This means that, even when changing the name is possible, users do not change their name often.

We can identify three situations for the change of name. Firstly, users want anonymity and thus, they change their name to avoid being recognized by other users. We can consider this as a "cheap" way to achieve anonymity, since it is actually cheaper than creating a new account, and the user does not have to re-classify her new account into a top-league (actually, quite expensive in terms of play-hours). Since changing the name does not require a change in the account, even with a new name users are actually easily recognizable. We will discuss this in the following section. Secondly, users may join a *team*. Teams are groups of avatars that frequently play together collaboratively against other teams. Teams also have associated names which users usually add to their names as a suffix. Thirdly, users may change their name by any other reason.

It is clear that, when finding two candidate avatars for merging, we cannot rely simply in comparing their names. On the one hand, very popular names such as "Batman" or "Superman" may be used by several users. On the other hand, even if a user has different accounts, and we successfully identify them for merging, nothing forbids the user to use different avatar names for those accounts. Thus, names will be used as *weak indicators* for avatar merging.

2) URL: In *Starcraft 2*, as described in Table III, an avatar is associated with a unique account. This account can be identified by the URL associated to the avatar which contains information about the location of the avatar (European Union, USA, Korea, etc.), the ID number and the avatar's name. As we have discussed, users are free to change the name of their avatars by paying a fee. When this is done, the URL changes by removing the old name and including the new one. However, since it is the same account, the location and the ID_number remain the same.

The URL is a *strong indicator* for avatar merging since it is quite obvious that, given two avatars with the same associated account, they correspond to the same user. Usually, we would integrate the traces of these avatars into a single one before the trace classification step (and actually, we do this for the first of our experiments). Instead, we will leave them as they are since they provide us with a sort of "ground truth". That is, if our system is able to merge avatars of different accounts, it should be able to merge avatars of the same account.

3) Surrogate Avatars: Given the set of traces T and the set of avatars A, we generate a partition of A in two different subsets A_γ and A_θ (a partition means that $A = A_\gamma \cup A_\theta$ and $A_\gamma \cap A_\theta = \emptyset$). For each $a \in A_\theta$, we generate a partition in T_a with components T_{a^1} and T_{a^2} where a^1, a^2 are called the surrogate avatars of a. Let $\tilde{A}_\gamma$ be the set of all surrogate avatars, we build the set $\tilde{A} = \tilde{A}_\gamma \cup A_\theta$.

Intuitively, surrogate avatars are known to belong to the same user. Thus, they provide a "ground truth" to evaluate our approach.

In Figure 4 the chart at left shows the long-tail distribution of the number of games played by avatar. We assume that professional players (those we are looking to disambiguate) are those that belong to the head of the curve, i.e. those that play the most. We consider this assumption fair since, in order to become professionals, players have to practice and perform in several competitions yielding a high number of games played. Thus, the set A_γ is built from a fraction γ of the avatars with the highest number of games played.

Similarly, we consider a minimal number of games (traces) to actually include the avatar a in A_θ. We assume that a user that has played a few games with an Avatar has no reasons to create a different one. We are well aware that this may not be the case for users that *already* have a different avatar and are just starting with their second. However, as we will discuss next, this induces an "imbalance" issue that may affect the classifiers' ability to group avatars of the same user. For an avatar with a few traces, the classifier will fail to provide a good prediction and the confusion matrix will present values which are explained by randomness rather than by the fact that two avatars belong to the same user. For example, consider an avatar a such as $|T_a| = 2$. Its row in $\tilde{C}^\rho$ will contain two non-zero values 0.5 in two different columns. It is easy to see that this avatar would likely conform a pattern concept with the avatars corresponding to those columns. In order to avoid this, we use a threshold θ such as for all $a \in T_\theta$ we have $|T_a| \geq \theta$.

The problem of balance refers to the fact that we are not certain about the distribution of time spent by a user among her different avatars. Put more simply, given that a user has

two avatars, the question is if she prefers one over the other (meaning that one of her avatars has more traces associated than the other) or if she plays equally with one or the other (meaning that both of her traces have a similar number of traces associated). This is an important issue since it affects directly the efficacy of our approach. If in general, users play many more games with one of their avatars than the others, the classifier applied to the traces will be less effective and will tend to classify the traces of the avatars with fewer games on the avatar with more games. To study this issue in a deeper way, we introduce the parameter β as a balance between the traces distributed over the surrogate avatars. Consider that for an avatar a we have that $|T_a| = 100$, this is the avatar has 100 associated traces. When creating the surrogate users a^1 and a^2 a $\beta = 0.5$ yields that both surrogate users will have 50 associated traces, i.e. $|T_{a^1}| = |T_{a^2}| = 50$. With $\beta = 0.7$ we will have $|T_{a^1}| = 70$ and $|T_{a^2}| = 30$ and so on.

B. Evaluation Metrics

To evaluate our approach we will measure the precision, recall and f-measure of the first 100 ranked avatar clusters. Given the ranking r (after cluster score filtering using λ), we have:

$$precision(r) = \frac{TP}{TP + FP}$$
$$recall(r) = \frac{TP}{TP + FN}$$
$$F\text{-}measure(r) = \frac{2 \cdot precision(r) \cdot recall(r)}{precision(r) + recall(r)}$$

Where TP, FP and FN stand for true positives, false positives and false negative, respectively. We will consider true positives as combinations of avatar names ($NAMES$), URL and surrogate avatars (SUG). False positives will be any candidate pair which does not belong to the true positive set in a given ranking. It is worth noticing that a pair considered as a false positive under this definition may not actually be one. We consider them false positives since we do not have enough information to consider them as true positives, meaning that their avatar names do not match, their URL is different and they are not part of our own set of surrogate avatars. *They are in fact the kind of pairs we are looking for*. False negatives are those candidate pairs that should have been considered as true positives, but that do not appear in the ranking.

The figure at left in Figure 3 shows the initial candidate pairs extracted from a confusion matrix generated by a Sequential Minimization Optimization (SMO) classification algorithm implemented in Weka. The classifier parameters were left as default, while the parameters of our approach for this particular figure are $\gamma = 0.05$ (top 5% of users were converted into surrogates) and $\theta = 5$ (users with less than 5 games were extracted from the dataset). Within the figure, a point represents a pair of avatars. If the avatars are surrogates, the point is represented with a red circle. If they have the same account, the point is represented with a green triangle and, in the case they have the same name, the point is a yellow star. In any other case (false positive), the point is a blue cross. False positives are annotated with the nick-names of the avatars. Figure 3 shows the top 20 without cluster score filtering ($\lambda = 0$) and presents very bad results. Only 8 out 20 points are not false positives (40% of precision). The figure at right in Figure 3 shows the top 20 after cluster score filtering ($\lambda = 0.9$) with very good results. Actually, only 1 out of 20 points is a false positive and it represents a couple of avatars that belong to the player known as `aLive`[4]. It is clear that in this particular case, our system is able to provide a precision of 100%, even though we just report 95% (19/20).

We also report on other three measures, namely P@10 (precision in the first 10 elements of the ranking), mean average precision (MAP), the receiver operating characteristic (ROC) (and the ROC area under the curve - AUC). For the sake of brevity, we do not provide a description of them. For further information on these metrics we refer the reader to [10].

V. Experiments

This section reports a thorough evaluation of our approach, for answering the *avatar aliases identification problem*. We begin by introducing our datasets and by highlighting the prediction ability of game traces when there are no aliases in the dataset.

A. Rough replay collections

Any game of *Starcraft 2* is recorded into a file called *replay* which contains all data necessary for the game engine to replay the game. Replays are shared on dedicated websites[5]. Along with replays, a set of parsing tools which allows extracting information from the replays are openly available[6]. Using these tools we have created two collections for our study.

Collection 1 – *Replays without avatar aliases:* This collection has been chosen for studying the efficacy of classifiers to recognize avatars of traces. Thus, we have purposely selected a collection of game replays which cannot contain avatar aliases. This is the case for the 2014 World Championship Series (second season[7]) in which users are forced to register their real names. The collection contains a total of 955 one-versus-one high level games and 171 unique players.

Collection 2 – *Replays with possible avatar aliases:* We gathered all the replays available on the website *Spawning Tool*[8] on the month of July 2014, for a total 10,108 one-versus-one games and 3,805 players. This collection corresponds to a real world situation, and is used for evaluating our avatar alias resolution approach according to Section IV. Figure 4 shows the distribution of the number of games by avatar for both replay collections. The distribution for Collection 2 corresponds to a long tail where 10% of players participate in more than 67% of the games. The distribution for Collection 1 is explained by the elimination process of the WCS qualifiers, meaning that the distribution of game played by user gradually increments as they go up the classification ladder. Respectively, in average each player participates in 5 and 9 games in Collections 1 and 2. Charts at center and right in Figure 4 illustrate the proportion of each type of action used as feature for avatar classification in our approach. Particularly, these figures show the actions for the first ten and thousand seconds

[4] http://wiki.teamliquid.net/starcraft2/ALive
[5] http://wiki.teamliquid.net/starcraft2/Replay_Websites
[6] http://sc2reader.readthedocs.org/
[7] http://wcs.battle.net/sc2/en/articles/wcs-2014-season-2-replays
[8] http://spawningtool.com/

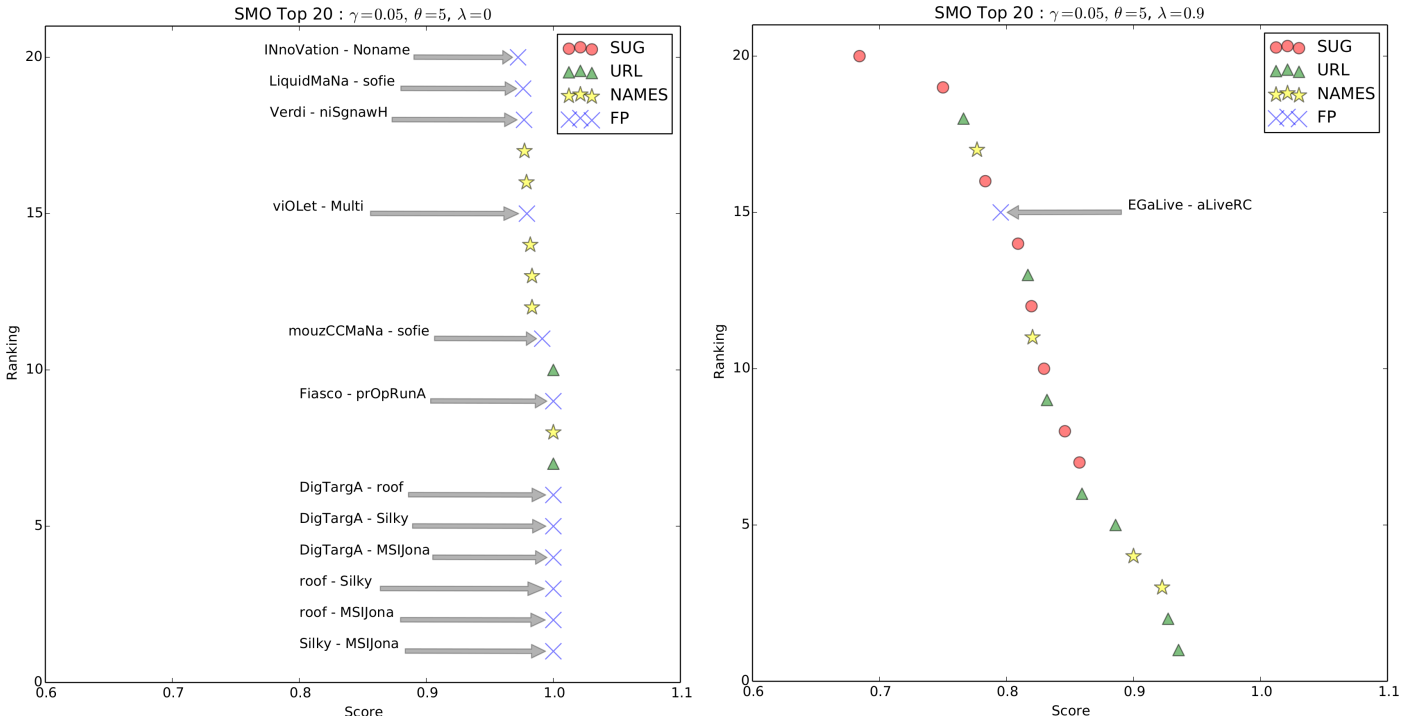

Figure 3: Candidate pairs ranking with $\lambda = 0$ (left) and with $\lambda = 0.9$ (right)

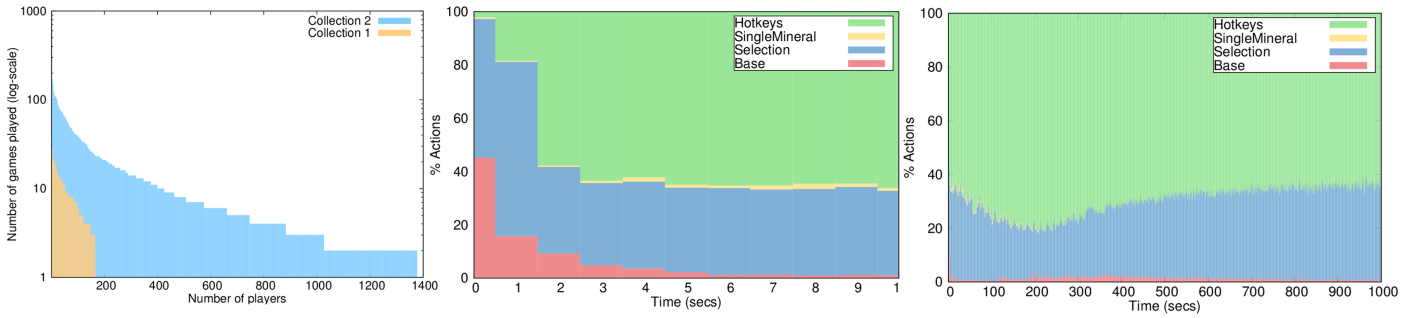

Figure 4: Distribution of the number of games by avatar, proportion of executed actions for first ten (resp. thousand) seconds

of the game, respectively. The object selection is the most prominent event in the first seconds of the game (the *warm up* phase), totalling 80% of the actions, after which the use of *hotkeys* becomes the most important action. This is the main reason why we will use object selection frequencies as features (in previous work, only hotkeys were used as features [21]). The chart at right in Figure 4 can be explained as follows. In the first minutes of the game there are not many options for the player to execute leading to a high proportion of clicking and selection events. After the 2 or 3 minutes, the user has built up a wider variety of options to execute which leads to hotkey bindings. The major part of bindings are made only once in the game, thus the highest proportion of hotkey events around 200 seconds. In the rest of the game, these proportions stabilize.

B. Experimental set up

Two main experiments were conducted. In Collection 1, we apply a set of classifiers to test the efficacy of predicting the avatar of a trace. In Collection 2, we apply classification and clustering to perform avatar alias resolution. Both experiments share the steps of (i) parameter selection, (ii) dataset creation and (iii) classification. The second experiment also considers a fourth step (iv) clustering, scoring and post processing.

1) Parameter selection: Throughout this document we have described a range of parameters which compensate for the fact that we know very little of the users we are looking for. For example, the parameter γ compensates for the fact that we do not know which is the proportion of avatars in a dataset that correspond to aliases of the same user. The parameter β compensates for the fact that we do not know in which proportion aliases are used. For example, $\beta = 0.5$ represents the fact that users may use their aliases in an equal proportion, while $\beta = 0.8$ represents the fact that users may use one alias much more than the other. The parameters in our approach are a manner of discovering under which conditions the notions of avatar alias resolution holds. For this reason, we have selected a wide spectrum of parameter combinations for our experiments. The following list present the names and meaning of each parameter.

τ: a time threshold for traces. Only the actions of the first τ seconds are considered in the dataset for classification. We also use a threshold for the number of actions, thus only the first τ actions can be considered
γ: proportion of users converted into *surrogates aliases*
θ: minimum number of games played by an avatar to be retained in the dataset
β: surrogates' balance is the proportion of games attributed to one of the two surrogates derived from an avatar
λ: cluster score threshold (see Section III)

2) Dataset creation: Having defined a set of parameter value, we generate datasets for classification from Collections 1 and 2. Each dataset contains traces as instances and avatars as class labels. Features of traces are a vector of numerical attributes and a couple of categorical attributes. Vector dimensions are associated with a canonical order over the repertory of events in the game. The value of each dimension for a given trace is the number of times the event was executed in the trace. A final dimension considers the average actions per minute (APM) associated with that trace. Categorical attributes correspond to the *race* used by the player (possible values: Protoss, Terran or Zerg) and the final status of the game (possible values: Win or Lose). Datasets are stored in the attribute-relation file format (ARFF) for the Weka system.

A single dataset is built for each different provided selection of attributes. For collection 1 we created 92 datasets, while for collection 2 we created 64 datasets.

3) Classification: Each dataset is classified using the Weka machine learning software and evaluated using 10-fold cross validation from which we obtain a confusion matrix. To represent the generality of our approach, we chose four different classifiers, namely K Nearest Neighbours (KNN), Naive Bayes (NBAYES), J48 decision tree (J48) and Sequential Minimization Optimization (SMO). Parameters for each of the classifiers were left as default.

4) Clustering, scoring and post processing: Each confusion matrix was processed by the Sephirot addIntent implementation[9] to obtain a set of pattern concepts. Scoring and post processing were implemented in ad-hoc python scripts.

C. Experimental results

1) Classifying avatars: Figure 5 shows the precision and the ROC area obtained for 92 datasets created for Collection 1. The parameter τ ranged over 23 values in an exponential scale, initially from 10 to 90 seconds then from 100 to 900 and finally from 1000 to 5000 seconds (the longest game in this collection has around 5300 seconds) and thus, the x axis of each figure is in logarithmic scale. For each measure, four figures corresponding to four different settings of θ are presented. Each line corresponds to a different classifier.

The figures present an empirical evaluation that the initial assumption, that avatars are very easily recognizable based in the signatures left in the traces they generate while playing, is true. For each different setting, ROC area is always around 100% showing the robustness of the approach under different

[9]https://code.google.com/p/sephirot/

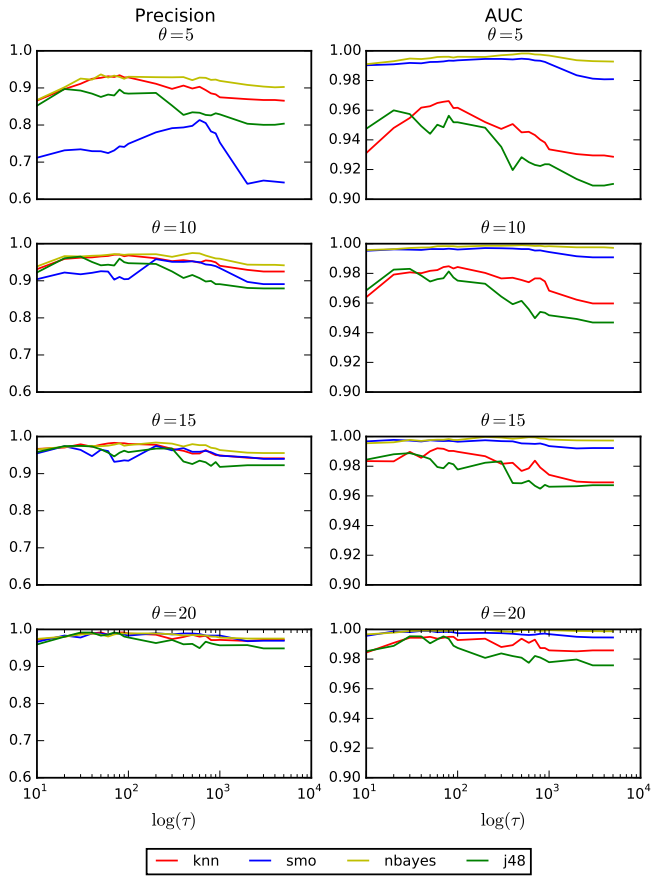

Figure 5: Classification results for Collection 1: precision and ROC area under the curve (AUC) distribution on 23 points of τ for four θ values. τ points were varied exponentially (10-90, 100-900, 1000-5000).

parametrizations. Precision is always maintained over 60%, achieving it minimal value for the SMO classifier with $\theta = 5$ and $\tau > 1000$. Actually, this also confirms two of our previous assumptions. Firstly, it is hard to recognize users that have played a few games, meaning that the larger the value of the θ threshold, more discriminative power has the classifier. Secondly, users are recognizable in the first few minutes of the game. The precision curves show a slight concave behaviour hinting a maximum of the precision w.r.t. the time cut used for traces. Indeed, this agrees with the hotkey use description given for Figure 4 in the chart at right. Users can be efficiently discovered by their hotkeys binding settings. As the game progress, traces may differ given that the number of options in the game greatly increase and vary in execution regarding different opponents.

2) Identifying multiple aliases: The goal of experiment 2 is evaluating our approach for finding avatar aliases. We have generated 48 datasets considering different parametrizations. As already discussed in the previous experiment, the efficacy of the classifiers achieves its best in the first few minutes of the game. Thus, we have selected three different τ values, namely $30, 60$ and 90 seconds. We have picked the same values for θ as in the previous experiments. Surrogates were generated for the first 5, 10, 15 and 20 percent of the most active users in the dataset (γ). For this particular experiment, we have set

the balance $\beta = 0.5$. All 48 datasets were processed using the four classifiers previously mentioned yielding a total of 192 confusion matrices. The top part of the Table IV shows a summary for the evaluation results using the top 100 pairs of avatar clusters found with parameters $\tau = 90$, $\theta = 20$ (248 avatars including surrogates), $\gamma = 0.2$ (41 surrogates) and $\lambda = 0.9$. The top, medium and bottom parts of the table contain the evaluations when looking for surrogates only, surragates and URLs, surrogates, URLs and names, respectively.

Results indicate that our approach is very efficient at identifying surrogate avatars under this parametrization. This is particularly true for KNN and the J48 classifiers achieving very high recall values. In the upper table, while precision is low it is worth noticing that in the top 100, there are only 41 surrogates meaning that the maximum achievable precision is 0.41. The classifier KNN is particularly good in this measure achieving an almost perfect value (0.4 of 0.41). All four classifiers achieve a very high precision in the first 10 results (P@10) while two of them get a perfect score. Indeed, one of the main characteristics of our approach is the good ranking it generates over the avatar pairs. This fact is confirmed by the good MAP and ROC area under the curve (AUC) values achieved by all four classifiers. Both these measures slightly degrade when including in the set of true positives URLs and names. In the case of the latter, this can be understood since not all avatars with the same name necessarily belong to the same user (as we have previously stated, same nick-names is a weak indicator). Thus, pairs of avatars with the same name will be more evenly distributed over the ranking or can even be found at the bottom indicating that they do not belong to the same user. This fact is reflected in the gap between the high grow of precision and low degradation of recall, i.e. avatars with the same name are evenly distribute between the pairs retrieved and those that were not.

A special mention deserve the URL true positives. As we have discussed, avatars with the same URL necessarily belong to the same user. Hence, we would have expected that in the first 10 pairs retrieved we could find an even distribution of surrogates and URLs. Instead, for all classifiers, P@10 is more than 80% surrogates (while the rest is always URLs - P@10 in the medium part of the table). The reason behind this is that we have purposely selected a balance of 0.5 for the surrogate distribution of traces, while we do not have control over this value for the URL pairs. The lower part of Table IV shows a summary of results when looking for just surrogates while varying the balance in the distribution of traces between them. We can clearly observe that the performance of the approach quickly degrades as more imbalanced gets the distribution (the higher the β value). Actually, for some classifiers it is not possible to obtain a single good result, even when we have lowered the λ threshold to 0.8. As URLs are not necessarily balanced, classifiers tend to predict the label of a trace belonging to an avatar with less traces to one with more traces. Issues related to learning from imbalanced datasets are reviewed in [7] and need to be considered when selecting a proper classifier for our particular application.

Parameters:: $\gamma = 0.2$, $\theta = 20$, $\lambda = 0.9$, $\tau = 90$						
Surrogates						
Classifier	F1	MAP	Recall	AUC	Precision	P@10
$j48$	0.468	0.824	0.805	0.904	0.33	1.0
$naivebayes$	0.226	0.740	0.390	0.915	0.16	0.8
smo	0.312	0.971	0.536	0.993	0.22	1.0
knn	0.567	0.822	0.976	0.882	0.4	0.9
Surrogates & URLS						
Classifier	F1	MAP	Recall	AUC	Precision	P@10
$j48$	0.588	0.907	0.606	0.866	0.57	1.0
$naivebayes$	0.443	0.857	0.457	0.864	0.43	1.0
smo	0.257	0.912	0.266	0.945	0.25	1.0
knn	0.670	0.937	0.691	0.874	0.65	1.0
Surrogates & URLS & Names						
Classifier	F1	MAP	Recall	AUC	Precision	P@10
$j48$	0.689	0.983	0.606	0.935	0.8	1.0
$naivebayes$	0.560	0.943	0.492	0.906	0.65	1.0
smo	0.258	0.949	0.227	0.960	0.3	1.0
knn	0.758	0.967	0.667	0.792	0.88	1.0
Parameters:: $\gamma = 0.2$, $\theta = 20$, $\lambda = 0.8$, $\tau = 90$						
J48						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.925	0.996	0.929	0.955	0.920	1.0
$\beta = 0.6$	0.545	0.927	0.632	0.921	0.480	1.0
$\beta = 0.7$	0.053	0.695	0.077	0.977	0.040	0.3
Naive Bayes						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.472	0.902	0.475	0.953	0.470	0.9
$\beta = 0.6$	0.273	0.923	0.316	0.973	0.240	1.0
$\beta = 0.7$	0.197	0.9	0.288	0.978	0.150	0.9
$\beta = 0.8$	0.048	0.533	0.120	0.983	0.030	0.3
SMO						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.392	0.983	0.394	0.992	0.390	1.0
KNN						
Balance	F1	MAP	Recall	AUC	Precision	P@10
$\beta = 0.5$	0.905	0.964	0.909	0.732	0.9	1.0
$\beta = 0.6$	0.750	0.957	0.868	0.929	0.660	1.0
$\beta = 0.7$	0.184	0.706	0.269	0.949	0.140	0.7

Table IV: Summary of evaluation measures over the resulting avatar cluster list yielded by the alias resolution approach (at top), and avatar clustering when varying the balance (β) (at bottom). Each entry represents a confidence matrix yielded by the respective classifier

VI. Related Work

Recently, Yan et. al [21] suggested that behavioural patterns discriminating skills, but also player themselves, can be discovered in the way they use their keyboard when playing *Starcraft 2*. They gathered 3,316 replays and took as features the frequencies of each control group key were used (30 features) for the whole game. They showed that these features allow predicting with high accuracy the league in which an avatar is playing[10]. A second result tells that a basic SVM classifier can predict the avatars involved in a game with high accuracy (≥ 0.95 accuracy with a leave-one-out validation), even when the avatar has few numbers of samples (between 2 and 20). Yan et. al showed that *hotkeys* (control groups) yield unique behavioural patterns of a user, but they did not present a way to discover avatar aliases: they even removed avatars with high probability of being aliases (e.g., *bar code names)*.

Using control groups as features is actually inspired by several works in software and security applications. Indeed, typing patterns allow identifying users by their typing characteristics [13]. The tedious task is to determine the appropriate behavioural metrics and features [20]. For example, keystroke dynamics and typing rhythm are crucial for authenticating a user based on habitual patterns [11]. Recently, an investigation around *Starcraft 2* [16] highlighted that the predictive impor-

[10]Leagues regroup players by level, following an ELO-like ranking system, from bronze, silver, gold, platinum, diamond, master, to grand master, the later involving the best 200 players of each continent.

tance of features is not constant across levels of expertise, while Yan et al. [21] ensured that complex features, even spatio-temporal, are not important: only the frequency of *hotkeys* is enough to output highly accurate classifiers; we emphasized this fact by showing that only the first few minutes of game play are enough to recognize the player.

Starcraft 2 and other real time strategy games (RTS) in general, face several research challenges in artificial intelligence [12] including *opponent modelling and learning*. Game traces/logs from replay files tend to be more and more used to tackle these problems since this information is easy and free to gather. We can notice several works focusing especially on tactical and strategic aspects, such as predicting army locations and opponents actions [19], [14] and automatically discovering build orders [9], [2]. All these works focus on effective actions made by players (build orders, micro/macro management) while we use here only the very first few actions of the *warm up* phase that one could consider as noise.

VII. Conclusion and Future Work

Video game analytics is a growing field of data science, crucial, if not vital, for the biggest game producers and editors. It comes with many challenges, the holy grail being to find all the ingredients that could assure an indisputable success of a game directly at its release. Pragmatically however, behavioural *big data* is gathered and analysed to answer several problems, including the design of better artificial agents, game balancing, bugs and cheaters and usurpers detection, etc. Games are then patched, cheaters are banned, and this cycle restarts. Behavioural data is also a gold mine in the context of electronic sports and competitive gaming, for reaching the same goals as in standard *sport analytics*[11].

We introduced the problem of avatar aliases identification, when there exists no mapping between individuals and their avatars. This is an important problem for game editors, but also for e-sport structures. Our method relies on the fact that behavioural data hide individual characteristic patterns, which allows making predictive approaches very accurate. Nevertheless, this good performance quickly degrades when data hides avatar aliases, which is why we based our analysis on confusion matrices.

Our results are encouraging, and suggest several perspectives of further development. Firstly, we assumed the mapping between players and avatars to be *one-to-many*. While this assumption is reasonable in the case of expert players in *Starcraft 2*, a general model suggests a *many-to-many* mapping in general. Secondly, we focused on expert players (having many historical samples), and it may be that predictive models are less powerful for novice players, since they use hotkeys in a smaller proportion. New features could be introduced in this case. For example, one could consider sequential features with sequential classifiers like in [4]. However, we believe that novice players do not hide behind aliases. How the method can be adapted for other video games is also challenging: the competitive games *Dota 2* and *League of Legends* do not propose customizable hotkey systems as in *Starcraft 2*, thus one needs to find good features from behavioural data. Finally, we proposed an approach in the context of formal concept analysis, but it is clear that other methods can be used and compared, relying on spectral clustering or biclustering.

[11] http://www.sloansportsconference.com

References

[1] Arnaud Giacometti, Dominique H. Li, Patrick Marcel, Arnaud Soulet. 20 Years of Pattern Mining: a Bibliometric Survey. *SIGKDD Explorations*, 2014.

[2] Guillaume Bosc, Mehdi Kaytoue, Chedy Raïssi, Jean-François Boulicaut, and Philip Tan. Mining balanced sequential patterns in RTS games. In *ECAI*, Frontiers in Artificial Intelligence and Applications, 2014.

[3] Gifford Cheung and Jeff Huang. Starcraft from the stands: understanding the game spectator. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 763–772. ACM, 2011.

[4] Gessé Dafé, Adriano Veloso, Mohammed Zaki, and Jr. Meira, Wagner. Learning sequential classifiers from long and noisy discrete-event sequences efficiently. *Data Mining and Knowledge Discovery*, 2014.

[5] Benhard Ganter and Rudolph Wille. *Formal Concept Analysis*. Springer, Berlin, 1999.

[6] Bernhard Ganter and Sergei O. Kuznetsov. Pattern structures and their projections. In *International Conference on Conceptual Structures (ICCS)*. Springer, 2001.

[7] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Trans. on Knowl. and Data Eng.*, 2009.

[8] Mehdi Kaytoue, Arlei Silva, Loïc Cerf, Wagner Meira Jr., and Chedy Raïssi. Watch me playing, I am a professional: a first study on video game live streaming. In *World Wide Web Conference, (Comp. Vol.)*, 2012.

[9] Cécile Low-Kam, Chedy Raïssi, Mehdi Kaytoue, and Jian Pei. Mining statistically significant sequential patterns. In *IEEE 13th International Conference on Data Mining*, 2013.

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval*. July 2008.

[11] Fabian Monrose and Aviel D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 2000.

[12] S. Ontanon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, and M. Preuss. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *Computational Intelligence and AI in Games*, 2013.

[13] Alen Peacock, Xian Ke, and Matthew Wilkerson. Typing patterns: A key to user identification. *IEEE Security & Privacy*, 2(5):40–47, 2004.

[14] Gabriel Synnaeve and Pierre Bessière. A bayesian model for opening prediction in RTS games with application to starcraft. In *IEEE Conference on Computational Intelligence and Games (CIG)*, 2011.

[15] T. L. Taylor. *Raising the Stakes : E-Sports and the Professionalization of Computer Gaming*. MIT Press, 2012.

[16] Joseph J. Thompson, Mark R. Blair, Lihan Chen, and Andrew J. Henrey. Video game telemetry as a critical tool in the study of complex skill learning. *PLoS ONE*, 2013.

[17] Dean van der Merwe, Sergei Obiedkov, and Derrick Kourie. AddIntent: A New Incremental Algorithm for Constructing Concept Lattices. In *Concept Lattices*. Berlin/Heidelberg, 2004.

[18] Arthur Von Eschen. Machine learning and data mining in call of duty (invited industrial talk). In *Machine Learning and Knowledge Discovery in Databases - European Conference, (ECML/PKDD)*, 2014.

[19] Ben George Weber and Michael Mateas. A data mining approach to strategy prediction. In *Symposium on Computational Intelligence and Games*, 2009.

[20] Roman V. Yampolskiy and Venu Govindaraju. Behavioural biometrics: a survey and classification. *International Journal of Biometrics*, 2008.

[21] Eddie Q. Yan, Jeff Huang, and Gifford K. Cheung. Masters of control: Behavioral patterns of simultaneous unit group manipulation in starcraft 2. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, 2015.