Predicting Personas Using Mechanic Frequencies and Game State Traces

Michael Cerny Green  
*Game Innovation Lab*  
New York University, Tandon  
New York City, USA  
mike.green@nyu.edu

Ahmed Khalifa  
*Institute of Digital Games*  
University of Malta  
Msida, Malta  
ahmed@akhalifa.com

M Charity  
*Game Innovation Lab*  
New York University, Tandon  
New York City, USA  
mlc761@nyu.edu

Debosmita Bhaumik  
*Game Innovation Lab*  
New York University, Tandon  
New York City, USA  
debosmita.bhaumik01@gmail.com

Julian Togelius  
*Game Innovation Lab*  
New York University, Tandon  
New York City, USA  
jan@togelius.com

Abstract—We investigate how to efficiently predict play personas based on playtraces. Play personas can be computed by calculating the action agreement ratio between a player and a generative model of playing behavior, a so-called procedural persona. But this is computationally expensive and assumes that a procedural persona is readily available. We present two methods for estimating play personas, one using regular supervised learning and aggregate measures of game mechanics initiated, and another based on sequence learning on a trace of closely cropped gameplay observations. While both of these methods achieve high accuracy when predicting play personas defined by agreement with procedural personas, they utterly fail to predict play style as defined by the players themselves using a questionnaire. This interesting result highlights the value of using computational methods in defining play personas.

Index Terms—game mechanics, machine learning, play persona, player modeling, videogames

I. INTRODUCTION

The use of personas has a long history within design in general and design for information technology in particular. The approach was pioneered for software development in the early 1990s [1] as a method for structuring and operationalizing qualitative data gathered from design research, chiefly in the form of interviews. Canossa and Drachen transported this approach into the realm of game design [2], defining personas less in terms of general life concerns and more in terms of player interaction preferences within the space of the game. They call their conceptualization *play personas* and operationalized their definition through data mining, suggesting how the persona design process could be supported by analyzing qualitative data gathered via telemetrics [3]. While play personas are archetypal models of player behavior inferred from experience or observed data, the re-projection into the game itself is something that is done imaginatively by the designer(s) of the game: i.e. play personas let us understand what players **have done**, but do not enact what players **might do**. **Procedural personas** [4]–[6] extend the play persona idea by adding a game-playing, generative aspect. By capturing persona characteristics from designer specification or from observed data, and formalizing these as utility functions, procedural personas are implemented as agents that can act in the game, enabling automatic playtesting.

In this paper, the use of personas comes full circle. Hand-designed utility functions steer artificial agents to behave in a specific way. These agents are then used to classify human playtraces with a ground truth label using Action Agreement Ratios (AAR) [6]. Though accurate, AAR calculations can be expensive for long playtraces. Agents must calculate their moves at every gametick. For tree-search and evolutionary agents, the allotted search time is typically proportional to their performance: the longer time given to search, the better they perform. Often times in the industry, game studios have hundreds to thousands of trace telemetry datapoints streaming in by the hour. For example, King, the developer of Candy Crush Saga, reports a daily average of 13 million active players\(^1\). For large or streaming datasets, calculating AAR can be an incredibly expensive computational challenge, even with parallelization. We propose training a machine learning algorithm on known playtraces to rapidly classify unseen playtraces with an acceptable level of accuracy compared to AAR. Classifying new playtraces does not require the use of agent play, so it has the potential to be much faster than AAR.

Additionally, we propose to use mechanic frequency vectors as an aggregated representation for the playtraces instead of game states. A mechanic frequency vector is an integer vector that represents the number of times each mechanic was activated during a playtrace. When compared to game state traces, a mechanic frequency vector is usually a smaller representation of a playtrace. To demonstrate the speed and accuracy of mechanic frequency vectors as a representation, we train classifiers on human playtrace data using AAR labels,

\(^1\)https://activeplayer.io/candy-crush-saga/
and infer on a testing set of more human playtraces using AAR labels. For comparison, we also train a classifier on the actual playtrace (sequence of game states) and compare its results to the ones trained on the mechanic frequency vector. Additionally, we train both classifiers on synthetic generated data using the same personas and on the human playtrace data labeled by humans themselves. While the classifiers trained on synthetic data and the classifiers trained on self-labeled human data fail to generalize to the test set, the classifiers trained on AAR labeled human data is able to accurately classify human playtraces.

II. BACKGROUND

The following subsections review game mechanic definitions and analysis, artificial persona research, and finally a brief description of the Minidungeons 2 game, within which this research was performed.

A. Game Mechanic Analysis

Though there is no universally accepted definition for what a “game mechanic” is, there are several popular definitions. Jarvinen [7] characterizes game mechanics as functional game features which allow the player to interact with the game state for achieving a desired goal state. Sicart [8] defines game mechanics as an action which is triggered by an agent to interact with the game world.

There are many ways to use mechanics to analyze player behavior and even generate content. Green et al. [9], [10] define “critical mechanics” as the set of game mechanics which need to be triggered to reach the win state. They use search based methods to find the critical mechanics to use as an input for automated video game tutorial generation. Silva et al. generate compact heuristics (i.e. mechanical instructions) for blackjack [11] and post-flop poker [12]. Khalifa et al. [13] generate Super Mario Bros (Nintendo, 1985) level snippets called “scenes,” which spotlight the use of mechanics to complete successfully, which Green et al. [14] build upon to develop entire levels in the Mario AI Framework [15] using “scene stitching.” Charity et al. [16] uses the quality diversity algorithm known as “Map Elites” to illuminate the mechanic design space for several video games within the GVGAI framework. Mechanic Miner [17] evolves mechanics for 2D puzzle-platform games, using Reflection to find a new game mechanic then generate levels that utilize it. With the the Gemini system [18], users can input mechanics to generate information about the game’s affordances [19]. Mappy [20] and the updated Mappyland [21] can transform a series of button presses into a graph of room associations, transforming movement mechanics into level maps for Nintendo Entertainment System games.

None of the above research proposes using mechanical frequencies as an aggregated representation of an entire playtrace, which is what we do in this paper. However, they do use mechanics to perform complex analysis on players, to provide content for players, and to estimate player behavioral patterns, which supports our proposed representation.

B. AI Personas

Video games allow players to experience levels designed by the games’ creators in almost an infinite number of ways. Each player has a distinct style, method, and strategy for navigating through a gamespace which we will refer to as a persona. Bartle [22] originally proposed a taxonomy of players and their personas based on how they interacted with the environment and other players ranging from killers, socializers, achievers, and explorers. Many game designers try to design their levels to cater to a specific persona or to design levels with multiple paths that work with different personas. Various attempts have been made to infer play personas through unsupervised learning from playtraces, including early work using data from Tomb Raider: Underworld [23] and Starcraft II [24].

Tychsen and Canossa [3] first introduced the concept of defining play personas through game metrics and mechanics performed. These metrics could be used to recreate a specific player but may not give as much insight as to what a type of player might do on a general scale (i.e. how would the same player react in a game level they have never played before.) With automated and artificial personas, these agents can be inserted into the game to predict how a player may approach a game level, saving human and computation resources and speeding up the design process. Procedural agents developed via evolution [25], [26] and reinforcement learning [27] have shown to be accurate in emulating player behaviors in a game setting. These artificial personas can be used for automated playtesting and encapsulate a variety of behaviors which lead to a wider diversity of levels designed for players with different play personas.

Action Agreement Ratio (AAR) is a metric of tracking behavioral similarity between playtraces proposed by Holmgaard et al [6], [28]. Generating AAR between a human and an agent is done by first reconstructing every game state of the human playtrace. At each state, the persona/agent being tested receives the state as input and calculates its next move. If the human action and the agent action are in agreement at that state, the agent’s agreement metric is increased by 1. After running over every state, the AAR is calculated by dividing the total score by the number of moves made. We use AAR as a ground truth in this paper to label human-based data.

C. MiniDungeons 2

MiniDungeons 2 2 is a 2-dimensional deterministic, turn-based, rogue-like game, first described in [29], in which the player takes on the role of a hero traversing a dungeon level, with the end goal of reaching the exit. Typically set on a 10 by 20 tile grid, the game map is made up of a mix of impassable walls and passable floors. Interactive items and characters are scattered throughout the level. To win, the player must reach the exit, represented as a staircase. All game characters have Hit Points (HP) and deal damage when the player collides with them. The player begins the game with 10 HP, and if the player runs out of HP, they die and lose the level.

2http://minidungeons.com/
Each turn, the player selects an action to perform, and all game characters will then move after the player completes their action. Any game character may move in one of the four cardinal directions (North, South, East, West) on their turn as long as the tile in that direction is not a wall. The player is given one re-usable javelin at the start of every level. The player may choose to throw this javelin and do 1 damage to any monster within their unbroken line of sight. After using the javelin, the hero must traverse to the tile to which it was thrown in order to pick it up and use it again.

While exploring a map, the player can find interactive objects that result in various effects:

- **Potions** increase the HP of the hero by 1, up to the max of 10.
- **Treasures** increase the treasure score.
- **Portals** come in pairs. Heroes can use to transport themselves to the paired portal on the same turn.
- **Traps** deal 1 damage to any game character moving through them, including monsters and enemies.

In addition to the above objects, the player may encounter monsters, all of which desire to attack a player within line of sight and some of which have additional secondary goals:

- **Goblins** move 1 tile every turn towards the player if within line of sight. They have 1 HP and deal 1 damage upon collision. Goblins try to avoid colliding with other goblins and goblins wizards.
- **Goblin Wizards** cast a 1 damage spell at the hero if they have line of sight within 5 tiles of the player. If they are over 5 tiles from the player but have line of sight, they will move 1 tile towards the player each turn. Wizards have 1 HP and deal no damage on collision.
- **Blobs** move toward either a potion or the hero within line of sight. They will move 1 tile towards the closest one per turn, preferring potions over the hero in case of a tie. A blob colliding with a potion or another blob consumes it and enables it to level-up into a more powerful blob. The default, lowest level blob has 1 HP and does 1 damage upon collision. The 2nd level blob has 2 HP and does 2 damage. The most powerful blob has 3 HP and does 3 damage.
- **Ogres** will move 1 tile towards either the player or a treasure per turn within line of sight, preferring treasures over the hero in case of a tie. When an ogre collides with a treasure, they consume it, and their sprite becomes fancier to look at. Ogres have 2 HP and deal 2 damage to anything they collide with, including other ogres.
- **Minitaurs** always move 1 step along the shortest path to the hero as determined by A* search, regardless of line of sight. Collision with the minitaur will deal 1 damage to the player. A minitaur has no HP and is immortal. When damaged, the minitaur is stunned for 5 moves.

### III. MiniDungeons 2 Playtraces

In a Minidungeons 2 gameplay session, players usually move around with their character to interact with different game objects and characters. At every frame, we record the current game state as a 2D map of tiles, the player’s current health, score, location, and last performed action. We also record all the interactions (game mechanics) that happens in the game when the player interacts with game objects and characters. Within the scope of this paper, there are a total of 17 mechanics tracked during play:

- **Enemy Kill:** the player slays any enemy
- **Monster Hit:** a specific monster is hit by either the player or the javelin. This mechanic is recorded separately for each different monster so we have **Goblin Hit**, **Minitaur Hit**, **Goblin Wizard Hit**, **Blob Hit**, or **Ogre Hit**.
- **Ogre Treasure:** the ogre collects a treasure
- **Blob Potion:** the blob consumes a potion
- **Blob Combine:** the blob combines with another blob
- **Javelin Throw:** the player throws the javelin
- **Collect Treasure:** the player collects a treasure
- **Consume Potion:** the player consumes a potion
- **Trigger Trap:** a trap is triggered
- **Use Portal:** the player uses a portal
- **End Turn:** the player makes a move and ends their turn
- **Die:** the player dies
- **Reach Stairs:** the player reaches the exit stairs

### A. Synthetic Dataset Generation

We curate a group of artificial agent personas to generate the synthetic playtraces. These agents are Best First Search (BestFS) agents which use utility functions created by Holmgaard et al [6], [26] in order to resemble a player archetype (persona). Previous research [26] uses MCTS agents, but in a fully deterministic game, BestFS generates more consistent results, both for action agreement and for gameplay ability. There are three agent personas in MiniDungeons that are used to capture agent playtraces with the following described goals:

1. **Runner:** complete the level as fast as possible.
2. **Monster Killer:** slay as many monsters as possible before completing the level.
3. **Treasure Collector:** open as many chests as possible before completing the level.

Agents perform online planning to move, meaning that each turn, they build a search tree to find the optimal next move to make. After doing preliminary experiments, we realized that the synthetic agents always make the same movements, and therefore multiple playtraces on the same map will not expand the dataset. To overcome this, we added random movement to the synthetic agents when generating the synthetic dataset: every move there is a 25% chance that the agent will take a random action rather than their searched action. This randomness is not used when performing action agreement.

At each turn, the BestFS agent uses the following heuristic for each different persona. In the following equations, $h_{agent}$ denotes the heuristic function for that specific BestFS agent. The agents could be runner ($r$), monster killer ($mk$), or treasure collector ($tc$). The runner agent is simply getting to
the exit in the fewest amount of steps as possible. Equation 1 shows the heuristic ($h_r$) function of the runner agent.

$$h_r = \text{dist}_{\text{exit}} - \text{steps}$$  \hspace{1cm} (1)

where $\text{dist}_{\text{exit}}$ is the distance from the current player location to the exit, and $\text{steps}$ is the amount of steps taken since the start of the game.

MK agents will prioritize killing all monsters while attempting not to die and getting to the exit when all monsters have been slain. Equation 2 shows the heuristic ($h_{mk}$) function of the monster killer agent.

$$h_{mk} = c \times N_{\text{monster}} + k \times \text{Dead} + p_{\text{monster}}$$

$$p_{\text{monster}} = \begin{cases} \min(\text{dist}_{\text{monster}}) & N_{\text{monster}} > 0 \\ \text{dist}_{\text{exit}} & N_{\text{monster}} = 0 \end{cases}$$ \hspace{1cm} (2)

where $c$ and $k$ are constants, $\min(\text{dist}_{\text{monster}})$ is the distance to the closest monster from the player location, $\text{dist}_{\text{exit}}$ is the distance between the player and the exit, $N_{\text{monster}}$ is the number of alive monsters in the level, and $\text{Dead}$ is a binary value that is equal to 1 if the player is dead and 0 otherwise.

TC agents will prioritize collecting treasure while attempting not to die and getting to the exit when all treasures have been collected. It uses a similar equation to 2 but replacing $\min(\text{dist}_{\text{monster}})$ with $\min(\text{dist}_{\text{treasure}})$ (the distance to the closest treasure from the player location) and $N_{\text{monster}}$ with $N_{\text{treasure}}$ (number of unopened treasures in the level).

### B. User Study

We prepare the following user study to gather human playtraces and to capture how users perceive their own play personas. The study is split into two parts, the first of these being a questionnaire and the second a gameplay session to gather playtraces. The questionnaire part consists of 10 multiple choice questions. For the questionnaire section, the first question of the study ("How often do you play games?") is to get insight into the player’s familiarity with gaming. Answer choices for this question are the following: Never, Once or twice a month, Once a week, A few times a week, Every day. This question is not specific about the genre or type of game being played, which can vary greatly.

The following list of questions contained within the gameplay questionnaire section of the user study are pertinent to the user’s self-perception of their play persona:

1. I try to finish the level (or a run of the game) as fast as possible
2. I look around a game area for all the hidden treasures or items before proceeding
3. I try to finish a level as move-efficient as possible
4. I try to defeat/kill all the enemies I can see
5. I enjoy battling enemies
6. I enjoy searching for hidden treasures or items
7. I only care about finishing the level
8. I make sure to collect every collectable object I can
9. I like fighting different kinds of enemies and collecting experience

Answer choices for the above questions are the following: Never, Rarely, Sometimes, Often, and Always. There are three questions which pertain to battling enemies, three questions about collecting treasures, and three questions about speed running. The purpose of these questions is to gain insight into the player’s perception of their own gameplay goals for dungeon-crawler, rogue-like games such as Minidungeons 2. Questions 2, 7, and 9 are inquiring about “runners”, questions 3, 6, and 8 about “treasure collectors”, and questions 4, 5, and 10 about “monster killers”.

After finishing the questionnaire, the player starts playing the game. There are 3 tutorial levels to acclimate the user to the game. These levels are very simple, beginning with a simple small map and slowly increasing in size and mechanical complexity until introducing all the basic mechanics by the 3rd level. The user is then presented with a series of levels randomly selected from the maps described in Section V-A.

### IV. MINIDUNGEON 2 LABELS

The collected playtraces in MiniDungeon 2 are labeled using different methods depending on the source of the data. Synthetic data is generated from the procedural personas, therefore their labels are already known. On the other hand, user data can be labeled either from the results coming from the questionnaire - the self-perceived label - or by comparing the user’s actions with the procedural agents’ actions - the action agreement label. While the synthetic data will always have a singular persona label, however the user data may have multiple play personas and thus can be labeled as multiple personas or possibly none at all.

#### A. Self Perceived Label

The self-perceived user playtraces labels are calculated from the answers the user selected for the questionnaire given at the beginning of the user study. These answers are transformed to numbers where Always is equal to 4 and Never is equal to 0. We average these across their related questions: questions 2, 7, and 9 are averaged under the “Runner” score; questions 3, 6, and 8 are averaged under the “Treasure Collector” score; and, questions 4, 5, and 10 are averaged under the “Monster Killer” score. At first, we decided to convert this average to a label by checking if the average value is greater than 2 (over 50%), in which case the associated persona is added to the user’s self-perception multilabel. We found that users tend to rank themselves high for most questions (section VI-A) causing more than 70% of all playtraces to be labeled Runner, Treasure Collector, and Monster Killer all at the same time. As a consequence, we decide to compare each average value to the average rating over all the user questionnaires as a form of normalization. If a participant’s score in a given persona category is greater than the average rating for that category over all questionnaires, the persona label is added to the participant. After performing this calculation, every playtrace of the participant is assigned the participant’s labels.
B. Action Agreement Label

The artificial persona agents described in Section III-A are used to generate action agreement labels for individual playtraces in both the human training set and the human testing set. This is done using a relatively simple replay process: Given a playtrace, a replay agent steps through every move that the human made. At every step, the system runs each of the 3 generative agents for 1 second, searching for the moves they would have made at that step. This means that each move takes 3 seconds to calculate agreement. User playtraces can be as long as 140 moves or more, meaning that a single playtrace could take several minutes to compute without parallelization. If the user’s action agrees with an agent’s action, that persona gets a point. The persona’s label is added to the playtrace if they agree more than they disagree with that persona, i.e. if they have more than 50% action agreement calculated by taking that persona’s point score divided by total steps taken.

V. EXPERIMENT

In this section, we describe our experiment setup in details. We begin with the MiniDungeons 2 maps that were selected and why. We then define our agent setup and finally explain the classifiers that are trained on synthetic and human playtraces.

A. Minidungeons 2 Maps

Five hand-designed maps were developed for this project. All maps contained a mix of game elements and offered multiple routes for the player to take to complete the level. Figure 1 shows all the five levels, as we can see all the maps except Map 202 provide very clear three paths for each different persona. For example, Map 100 has a center path for runner, right path for treasure collector, and left path for monster killer. Meanwhile, Map 202 is an open space with monsters and treasures distributed all over the map. Every map contains 5 – 6 monsters, 6 – 9 treasures, and at least one straightforward path to the exit.

B. Agent Setup

All agents are given 1.0 second to plan before making their next move. For the monster killer agent and treasure collector, we used $e$ equal to 45 and $k$ equal to $\text{double.max}$ to encourage the agent to stay alive and favor states that kill more monsters in case of monster killer or collect more treasure in case of treasure collector. Each persona runs 100 times each on the five map described in Section V-A, for a total of 1500 playtraces.

C. User Study Data

A link to the user study described in Section III-B was spread by a combination of tweets and academic email lists. 205 participants completed the whole study, while an additional 158 only played the game and did not answer the questionnaire. We were able to extract 565 playtraces to use as training set data among the participants. Our test data is made of additional 293 playtraces from participants, which brings our total dataset size to 853.

D. Classifying Players

Players can be classified as Runners (R), Treasure Collectors (TC), or Monster Killers (MK) using a machine learning model trained on labeled playtraces. We trained multiple different models and tested them on a common test set consisting of 293 playtraces that are labeled using AAR. The training dataset composed of 565 playtraces is divided into a 70-30 split for training and validation for all of the models.

We train our models on two different types of inputs:

- **Cropped playtraces**: We convert the states of the playtraces into a smaller state by cropping the states to focus on the 3x3 area around the player. This particular size was used because player interactions and the mechanics examined for the MiniDungeons game only affect the immediate surrounding area of the player (i.e. collecting nearby treasure or potions or attacking.) The smaller size also prevents the machine learning model from overfitting and ideally helps it generalize better [30].

- **Mechanic frequencies**: We use the frequency of the 17 triggered mechanics specified in Section III as the input for our model. By “frequency” we mean the exact number of times each mechanic is triggered over the course of one level. These mechanics are represented in vector format, and normalized on a mechanic-basis across the entire dataset before training.

These two input representations are used for all 3 datasets described in sections III and IV:

1) The synthetic agent dataset with labeled personas.
2) The user dataset with self-perceived labels calculated from the questionnaire.
3) The same user dataset as above but labeled by Action Agreement measured using the previously mentioned synthetic personas.

Based on these different input data representation, we decided to train two different types of models:

- **Long Short Term Memory (LSTM) Model**: this model is trained on the cropped playtraces. LSTM can train and generalized on sequential data. The model consists of a LSTM layer with 100 hidden nodes followed by an output layer of 3 nodes for each persona probability. The LSTM model is trained using stochastic gradient descent for 200 epochs and learning rate of 0.001. For each experiment, we train 3 networks to make sure the results are stable.

- **Support Vector Machine (SVM) Model**: this model is trained on the mechanic frequencies. We picked SVMs because they are a well established machine learning method that is known for its capability to generalize on small datasets without overfitting.

VI. RESULTS AND DISCUSSION

In this section, we review the results of the user study questionnaire detailed in Section III-B and discuss the results of the classifier training explained in Section V.
A. User Study Questionnaire

The responses to the question “How often do you play video games” demonstrate that over 89.3% (184) of participants play video games once a week or more. We can assume from this that the vast majority of participants are at least familiar with optimizing gameplay actions for the purpose of achieving some goal, such as winning.

Figure 2 shows the results for all these questions and their averages. The numbers/color in each cell corresponds to the number of users with that response. We can see that most users answer Always, Often and Sometimes, suggesting that they see themselves as a mix of different personas rather than one in particular. Out of the 206 participants, 154 perceived themselves to be Runners, 151 perceived themselves to be Treasure Collectors, and 159 perceived themselves to be Monster Killers, according to their self-perception multilabel.

B. AAR and Self-Perceived Label Analysis

Table Ia displays the breakdown of player distributions using every AAR multilabel combination. One interesting finding is that there is a sizable group of playtraces (82) with no label at all, suggesting the presence of a fourth persona not measured here. Another curious group of players are the 117 traces that are classified as all three persona types. The mechanics showcased here are selected due to their traditional correlation with a specific persona, and values in bold correspond to that combination’s correlated mechanics.

All combinations which include the “R” label contain a lower amount of steps taken than other label combinations, the lowest coming from “Pure R” at 18.49 and “R&MK” at 15.25. “Pure TC” and “Pure MK” distributions contain similar amounts of treasures collected and monsters killed, but there are only 2 “Pure MK” playtraces in the entire dataset. The presence of many “R&MK” (56), “TC&MK” (24) and “R&TC&MK” playtraces but few “Pure MK” traces promotes the idea that players that are driven to kill monsters are also driven toward other persona subgoals during gameplay. The highest combinations of treasures collected and enemies killed comes from the “TC&MK” labeled traces, 6.83 and 5.0 respectively, suggesting TC and MK subgoals have great synergy. The “R&TC” traces have a greater amount of enemies killed than “R&MK,” but the treasure collection amounts differ greatly (3.36 and 0.18 respectively). Due to the nature of enemy movement in Minidungeons 2 and the placement of enemies and treasure on some of the user study maps (Section V-A), it is difficult to collect treasure and not kill monsters. However, it is quite simple to kill monsters and not collect the nearby treasure, which is immobile. We think that “R&TC” players will kill monsters on the way to collect treasure, whereas “R&MK” players kill monsters and ignore the treasure. Also of note is the different in steps taken between the two distributions: “R&TC” players (36.82) take over twice as many steps as “R&MK” players (15.25) in order to collect more treasure. This result suggests that “R&TC” players may have competing subgoals: “go to the exit as fast as possible, but also go out of your way to collect treasure.”

Table Ib displays the breakdown of player distributions using the labels coming from the questionnaire answers for all multilabel combinations. We notice that more than 60% of all the playtraces are either ‘No Label’ or ‘R & TC & MK’. Looking at the different distributions of ‘Steps Taken’, ‘Treasure Collected’, and ‘Enemies Killed’, we noticed that any combination with runner persona have the lowest number of ‘Steps Taken’ compared to all the other similar to AAR labels. We think that runner players have a high tendency to not change their persona compared to others. On the other hand, we think other personas change themselves to become a runner while playing the game. This can be seen by comparing the number of playtraces that are classified as runner by AAR label compared to the self perceived labels. Another noticeable thing is the distributions for all the personas have a higher standard deviations compared to the AAR labels which confirms our point that users are either changing their personas
during playing or can’t identify themselves correctly. Finally, both ‘Treasure Collected’ and ‘Enemies Killed’ averages are way lower than the ones shown by the AAR labels. This suggests that users who think themselves as a monster killer or a treasure collector actually behave more like a Runner, as the mean of ‘Steps Taken’ mechanic is low by comparison.

The questionnaire data presents a rare opportunity to compare human self-perception with a synthetically generated ground truth. To gauge the accuracy of self-perception, we compare the self-perceived label with the multilabel generated from action agreement. We calculate the amount of times that both labels either said a playtrace was a persona or both said a playtrace was not a persona. Self-perceived and action agreement labels both agree (both are “Yes” or “No” at the same time) on Runner labels 47.3% of the time, Treasure Collectors 59.1% of the time, and Monster Killers only 50.4% of the time. For all three persona styles, agreement is a coin toss. The confusion matrices (Table II) between AAR and Self-Perceived labels demonstrates this disagreement across each of the persona types.

The discrepancy between perception and action agreement labels suggests a possible cognitive dissonance between how players believe that they play and how they actually do play. Another possible reason for this disagreement is that users believe that they play and how they actually do play. Players can change their play persona to adapt to any factor be it environmental rewards (the level might be easier or more enjoyable for a treasure collector persona over another) or agential motivations (the player is bored and decided to race through the level like a runner). Lastly, the participants of this study may be more familiar with other games in which they behave how they perceive themselves to behave. Since Minidungeons 2 is a new environment for them, their lack of familiarity may have caused them to behave differently.

### C. Classifier Training and Testing

Table III displays the results of training and validation of the classifiers on the synthetic and human datasets. The SVM trained on synthetic data clearly fail to capture the distribution of the human players. Although there are 1,500 playtraces to train on with different distribution, the playtraces distribution is far away from human data distribution. We think due to the purity of the synthetic agents (only runner, only monster killer, or only treasure collector), the trained classifier fails to recognize any multi-label persona. As a result, they do not perform well on the human testing set with an accuracy of 4.8%. The LSTM model is also not able to generalize well on the synthetic data, having a low accuracy during training (58.1%) and testing (18.6%). During training, the LSTM is not able to differentiate between treasure collector and monster killer personas (due to proximity of treasures and monsters) which causes it to classify all the treasure collector as monster killers during training. We believe that the lack of variety between playtraces explains this poor performance for both the LSTM and the SVM.
mechanical patterns that the classifier picked up on. The similarity in scores between LSTM model and SVM model supports our hypothesis that mechanic frequency is a good data augmentation method, being able to capture the essence of the play persona similar to using playtrace data.

VII. CONCLUSION & FUTURE WORK

In this paper, we showcased that training a machine learning model on user data that is labeled using action agreement ratio is capable to generalize and predict the play personas of unseen players. We use two different input representations: a cropped sequence of states and mechanic frequencies, showcasing that both can be used interchangeably depending on what is easier to capture in the game. We demonstrate that just using synthetic data from the agents or self-labeled playtraces fails to generalize well. By modeling with a small human AAR labeled dataset, we can successfully predict the personas of unseen human playtraces. Game states and mechanic frequencies are problem independent, meaning that all they require is the user to define what a state or a mechanic is as well as build a classifier that can take them as input. These are usually defined during the game design process anyways and typically do not require more work from developers.

Calculating the actual labels using AAR took on average 40 seconds per playtrace with parallelization and almost 7 hours for the 8 trace training set in entirety. For comparison, training the SVM on mechanic frequency vectors took less than 5 seconds overall and can infer on new data in less than a second. Time is not usually a concern during development. But in published games with millions of active players playing daily it becomes infeasible to calculate play personas using AAR. At the speed at which we labeled the 853 trace dataset using AAR, labeling of a 10,000 trace dataset takes 4.5 days.

The importance of identifying persona goes beyond just testing if the levels have the required experience but also generating personalized levels [14] and/or tutorials [10]. Outside of PCG, play persona statistics can be a valuable source of feedback for designers to improve and iterate levels. Play personas can easily be integrated in the development engines as tool to direct designers about map and level experience.

ACKNOWLEDGMENT

We would like to thank those that took part in our user study and took the time to answer the questionnaire.

REFERENCES

[1] A. Cooper, “The inmates are running the asylum: Why high-tech products drive us crazy and how to restore the sanity (new ed.),” Indianapolis, Ind.: Sams, 2004.

[2] A. Canossa and A. Drachen, “Patterns of play: Play-personas in user-centred game development,” in Breaking New Ground: Innovation in Games, Play, Practice and Theory: Proceedings of the 2009 DiGRA Conference, London, Brunel University (September 2009), 2009, p. 11.

[3] A. Tynchen and A. Canossa, “Defining Personas in Games Using Metrics,” in Proceedings of the 2008 Conference on Future Play: Research, Play, Share. ACM, 2008, pp. 73–80.

[4] C. Holmgård, A. Liapis, J. Togelius, and G. N. Yannakakis, “Evolving models of player decision making: Personas versus clones,” Entertainment Computing, 2015.

[5] ——, “Generative agents for player decision modeling in games,” in Proceedings of the Foundations of Digital Games Conference, 2014.

[6] ——, “Personas versus clones for player decision modeling,” in International Conference on Entertainment Computing, 2014.

[7] A. Järvinen, Games without frontiers: Theories and methods for game studies and design. Tampere University Press, 2008.

[8] M. Sicart, “Defining game mechanics,” Game Stud., vol. 8, 2008.

[9] M. C. Green, A. Khalifa, G. A. Barros, and J. Togelius, “Press space to fire: Automatic video game tutorial generation,” in Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference, 2017.

[10] M. C. Green, A. Khalifa, G. A. Barros, T. Machado, A. Nealen, and J. Togelius, “Atdelfi: automatically designing legible, full instructions for games,” in Proceedings of the 13th International Conference on the Foundations of Digital Games, 2018, pp. 1–10.

[11] F. de Mesentier Silva, A. Isaksen, J. Togelius, and A. Nealen, “Generating heuristics for novice players,” in 2016 IEEE Conference on Computational Intelligence and Games (CIG), 2016, pp. 1–8.

[12] F. de Mesentier Silva, J. Togelius, F. Lantz, and A. Nealen, “Generating novel heuristics for post-flop poker,” in 2018 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, 2018, pp. 1–8.

[13] A. Khalifa, M. C. Green, G. Barros, and J. Togelius, “Intentional computational level design,” in Proceedings Of The Genetic and Evolutionary Computation Conference, 2019, pp. 790–803.

[14] M. C. Green, L. Mugrai, A. Khalifa, and J. Togelius, “Mario level generation from mechanics using scene stitching,” in 2020 IEEE Conference on Games (CoG). IEEE, 2020, pp. 49–56.

[15] J. Togelius, S. Karakovskiy, and R. Baumann, “The 2009 mario ai competition,” in IEEE Congress on Evolutionary Computation. IEEE, 2010, pp. 1–8.

[16] M. Charity, M. C. Green, A. Khalifa, and J. Togelius, “Mech-elites: Illuminating the mechanic space of gvg-ai,” in International Conference on the Foundations of Digital Games, 2020, pp. 1–10.

[17] M. Cook, S. Colton, A. Raad, and J. Gow, “Mechanic miner: Reflection-driven game mechanic discovery and level design,” in European Conference on the Applications of Evolutionary Computation, Springer, 2013, pp. 284–293.

[18] A. Summerville, C. Martens, S. Harmon, M. Mateas, J. Osborn, N. Wardrip-Fruin, and A. Jhala, “From mechanics to meaning,” IEEE Transactions on Games, vol. 11, no. 1, pp. 69–78, 2017.

[19] M. Mateas, “A preliminary poetics for interactive drama and games,” Digital Creativity, vol. 12, no. 3, pp. 140–152, 2001.

[20] J. Osborn, A. Summerville, and M. Mateas, “Automatic mapping of nes games with mappy,” in Proceedings of the 12th International Conference on the Foundations of Digital Games, 2017, pp. 1–9.

[21] J. C. Osborn, A. Summerville, N. Dailey, and S. Lim, “Mappyland: fast, accurate mapping for console games,” in Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 17, no. 1, 2021, pp. 66–73.

[22] R. Bartle, “Hearts, clubs, diamonds: Players who suit muds,” Journal of MUD research, vol. 1, no. 1, p. 19, 1996.

[23] A. Drachen, A. Canossa, and G. N. Yannakakis, “Player modeling using self-organization in tomb raider: Underworld,” in 2009 IEEE symposium on computational intelligence and games., IEEE, 2009, pp. 1–8.

[24] T. Avontuur, P. Spronck, and M. Van Zaane, “Player skill modeling in starcraft ii,” in Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 9, no. 1, 2013.

[25] C. Holmgård, A. Liapis, J. Togelius, and G. N. Yannakakis, “Evolving personas for player decision modeling,” in 2014 IEEE Conference on Computational Intelligence and Games, IEEE, 2014, pp. 1–8.

[26] C. Holmgård, M. C. Green, A. Liapis, and J. Togelius, “Automated playtesting with procedural personas through mcts with evolved heuristics,” IEEE Transactions on Games, vol. 11, no. 4, pp. 352–362, 2018.

[27] C. Holmgård, A. Liapis, J. Togelius, and G. N. Yannakakis, “Generative agents for player decision modeling in games,” in Foundations of Digital Games, 2014.

[28] ——, “Monte-carlo tree search for persona based player modeling,” in Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference, 2015.

[29] C. Holmgård, J. Togelius, A. Liapis, and G. N. Yannakakis, “Minidungeons 2: An experimental game for capturing and modeling player decisions,” in Foundations of Digital Games, 2015.

[30] C. Ye, A. Khalifa, P. Bontrager, and J. Togelius, “Rotation, translation, and cropping for zero-shot generalization,” in 2020 IEEE Conference on Games (CoG). IEEE, 2020, pp. 57–64.