Categorical Clustering Applied to the Discovery of Character Builds in TCTD2: the BaT Approach

1st David Renaudie
Massive Entertainment, Ubisoft
Malmö, Sweden
david.renaudie@massive.se

2nd Robert Lizatovic
Massive Entertainment, Ubisoft
Malmö, Sweden
robert.lizatovic@massive.se

3rd Ahmad Azadvar
Massive Entertainment, Ubisoft
Malmö, Sweden
ahmad.azadvar@massive.se

4th Rickard Elmqvist
Massive Entertainment, Ubisoft
Malmö, Sweden
rickard.elmqvist@massive.se

5th Klaus Hofmeister
Massive Entertainment, Ubisoft
Malmö, Sweden
klaus.hofmeister@massive.se

6th Björn Kristmannsson
Massive Entertainment, Ubisoft
Malmö, Sweden
bjorn.kristmannsson@massive.se

***Abstract*—This article describes an attempt to categorize character configurations of players of *Tom Clancy's the Division 2*, conducted to highlight behavioral differences in approach to gameplay based on one's character build. Nine distinct character builds were extracted for maximum coherence and minimum variance and each build showed significant differences in separate measures of behavior such as playtime, character health and armor among other attributes. The proposed method was also able to recover builds recognized by social forums as well as discovering new ones. Appropriation of Character builds as categorical text-based data (BaT: Build as Text), provides a unique opportunity for game researchers to use a diverse set of input data which will in turn contribute to the improvement of the process of game design informed by player choices. Longitudinal observations in interconnection of obtained clusters may provide further insight into formation and evolution of gameplay types.**

***Index Terms*—Player Modelling, Behavioral Analysis, Categorical Clustering, Player behavior, Character Builds**

I. Introduction

Popularity of video games in the recent years has made the stream of player behavioral data more readily available [1]. On the other hand, advancements in technology and methods of analysis of such data has resulted in series of novel approaches to classify and further analyze player data for a myriad of purposes such as churn behavior and prediction [2], identification of social influencers [3], exploration of emerging game loops [4] and even AI agents with evolving personalities [5]. Classification of player data, other than player churn and retention behavior [6], could be used to increase player efficiency in gameplay based on the choices afforded by the game [7]. Such approaches could help game developers to improve gameplay features and help them personalize representation of options [8]. It may also diversify the understanding of players and their varied needs [9]. In this study, we introduce the BaT (Build as Text) approach to categorize player character configurations in Tom Clancy's the Division 2 (TCTD2;*Ubisoft, 2019*), an online multiplayer shooter. We explain the principle of our approach, that is encoding textual descriptors paired with applying Latent Dirichlet Allocation [10], and using Shannon Entropy [11] for allocation of data points to clusters. We show the advantage of the proposed method by comparisons made with conventional numerical clustering approaches. As character builds are highly reflective of playstyle, their categorization can be seen as a proxy to categorization of player behavior. Therefore, performance of the model in view of behavioral metrics such as health, armor, skill power, offense, defense, utility and overall playtime split is examined to show that the different discovered builds are indeed representative of specific character attributes and playtime behavior. Finally, we show that the BaT approach is efficient in discovering both known (predefined) and unknown (emergent) builds and further discuss implications, limitations and avenues of future work.

978-1-7281-4533-4/20/$31.00 ©2020 European Union

II. Background

Application of advanced statistical methods in processing data generated by player behavior in video games has been attempted before [2]–[6]. The current study argues that the nature of data generated by player behavior may provide the first clues about choosing the algorithm to treat them. In this case, Latent Dirichlet Allocation algorithm [10] was utilized for categorizing character configuration data from TCTD2, encoded in text-like format, which proved more suitable than conventional numerical approaches.

A. Behavioral modeling in Games

Player modeling is the study of computational models of players in games [12]. As video games provide the unique conditions of interactivity as well as affective simulation, the dynamic nature of player behavioral data could be utilized to gain insight into player preferences [12], understand the underlying motivations for play [9], quality of the experience [13], and to make adaptive games [14]. Other than business applications of player behavioral modeling conventionally focused on player churn behavior and spending habits [15], and experimental modeling for method development [16], design

oriented clustering studies on large scale samples of players of popular commercial video games are mostly focused on numerical clustering methods such as k-means [17] and are rarely conducted by industry practitioners [18] with context dependent, high dimensional dataset such as the current study [19]. Centroid-based clustering methods (e.g. k-means) have been successfully used to classify player behavior [20] due to their ease of use for large scale numerical values and low time complexity [21]. However, previous research [7] showed that they may not produce reliable results. Density based clustering methods, although employed less frequently, have also proven useful in modelling player behavior with reduced dimensionality [22]. But as we will discuss in Section III-A2, their dependence on the availability of pairwise distances between data points makes them impractical for handling large amounts of high dimensional data. Therefore, treatment of special data types has led practitioners and researchers to employ alternative methods. In one occasion [4] a sequence clustering method was used to identify and categorize game loops based on the frequency of repeated consecutive action types in a video game. In another example, [2] treatment of contextual time-series data was conducted with Dynamic Time Warping and a series of other refined algorithms for evaluation of game events. Other studies [31], [32], also used methods similar to those of the current study when dealing with text based data, although applied to player in-game chat logs or tweets about a game, in order to categorize and uncover player communications. Regardless of data type, the common thread in clustering studies is the emphasis on the importance of context not only in mining and interpretation of data [7] but also in the methods that make sense of relationships between these data points.

B. The Game

Tom Clancy's The Division 2 (TCTD2) is the second installment of an online multiplayer tactical shooter. The game is fully online, even for solo play. The client software is constantly communicating in-game telemetry to the game servers, where players can meet and join, for cooperative or competitive activities. Other than character's aesthetics, players can upgrade, unlock and modify their characters' attributes, skills and equipment. Character attributes are progression dependent values of baseline characteristics such as health, armor and skill power. Skills are special capacities, unlocked by player progression. Equipment (a.k.a. *loot*), such as weapons or armor pieces, is procedurally generated and found during in-game activities. As a consequence, character configurations are fine-tuned by players among a set of unique possibilities that has combinatorial complexity. The choices made in this superior cardinality space reveal player tendency towards archetypal playstyles, while leading to a clustering problem of high dimensionality.

C. Data

The dataset used in this study contained approximately 250000 character configurations. These configurations were obtained from a special 'player status' event that is sent by the game client on game start (plus a few other conditions that are not relevant here), using the built in telemetry system. For this study, we used event data collected over a period of 1 week (from 06.11.2019 to 13.11.2019). To reduce data volume, we selected only the latest player status event for each unique player ID, therefore keeping only the latest character configuration snapshot of each player that was active during that period.

The character configuration data point consists of features that are sufficient to specify a character in the game (see Table I) but do not describe explicitly behavior nor playstyle. These character features can broadly be grouped into 4 categories: attributes (health, armor, skill power etc.), skills (and skill variants), equipment (equipped weapons, gear etc.), and specialization (specialized equipment and skills that go together for a specific purpose. e.g. Sharpshooter specialization for a sniper). Some features are numerical, while others are categorical. The raw dataset contains 56 base features.

III. Clustering

A. Numerical clustering methods

As an initial approach for identifying builds, we tried using the more classical centroid-based and density-based clustering techniques on the character configurations.

1) Centroid-based clustering: For k-means clustering, all raw character configurations are encoded as numeric feature vectors. A normalization step is applied on all the numeric features to bring them on the same scale (0-1), whereas categorical features are one-hot-encoded (OhE). The dataset transformed in this manner contained over 7000 features, only 6 of which were non-binary. To reduce dimensionality, we apply truncated singular value decomposition (tSVD), projecting it onto a subspace of 200 dimensions. This operation retains about 70% of the variance in the encoded data. The elbow method is then employed using the silhouette coefficient [23] and the sum of squared displacements to determine the optimal number of clusters. Clusters are visualized using tSNE embedding [24] over a range of perplexity values.

2) Density-based clustering: Due to the necessity to precompute a custom distance matrix, we could only afford to perform density-based clustering on a small subset of the original data. Therefore, we randomly sample 12k character configurations from the original 250k. As in the case of k-means, numerical features are first normalized to a range of 0-1. However, as our dataset contains a mixture of numerical and categorical features, we need to use a distance metric that can handle both data types. One such metric is the Gower distance [24], which is a true metric and satisfies the triangle inequality. As such, it can be used with DBSCAN [25], but requires the pairwise distance matrix to be computed up front. To obtain the "best" (most compact) clustering results, we perform a grid search over the DBSCAN parameters epsilon and minimum number of points and maximize the silhouette coefficient. Results are then visualized using tSNE embedding over a range of perplexity values.

TABLE I
CHARACTER CONFIGURATION FEATURES

Feature category	Feature	Type[a]	Example	Description
Attributes	Armor	Numeric	100000	Armor rating
	Health	Numeric	200000	Health rating
	Skill power	Numeric	600	Effectiveness of skills
	Offense	Numeric	2	Overall offensive tier
	Defense	Numeric	13	Overall defensive tier
	Utility	Numeric	12	Overall skill tier
Skills	Skills	Categorical (multi - 2)	"Hive"	The 2 skills equipped
	Skill variants	Categorical (multi - 2)	"Hive - revive"	Modification of the base skills equipped
Equipment	Weapons	Categorical (multi - 3)	"assaultrifle_pof416_t2_v1"	Names of the 3 weapons equipped
	Weapon types	Categorical (multi - 3)	"Assault Rifle"	Types of weapons equipped
	Body slot items	Categorical (multi - 6)	"Diamondback gloves"	Names of the 6 body slot items equipped
	Item brands	Categorical (multi - 6)	"Gila Guard"	Names of the brands of the 6 equipped items
	Gear talents	Categorical (multi - 18)	"Clutch"	Names of the gear talents equipped (up to 18)
	Weapon talents	Categorical (multi - 9)	"Optimized"	Names of the weapon talents equipped (up to 9)
Specialization	Specialization	Categorical (single)	"Technician"	Character specialization
	Signature weapon	Categorical (single)	"Crossbow"	Signature weapon used

[a]Specifies whether the feature is numerical or categorical. In case of categorical features, the brackets specify whether they are single-valued or multi-valued (and how many values are allowed)

B. The BaT Approach

1) Principle: In an industrial context, the produced clusters have to be concretely used by game experts, having low to no knowledge of data science methods and tools. Ease of interpretation of the method's output is thus key to adoption of the overall solution.

The inherent limitations of numerical clustering methods lie in either the necessity for encoding categorical features in a numerical vector space (e.g. for k-means), or the need for specialized distance metrics that can deal with both numerical and categorical features - such as the Gower distance [24] for DBSCAN - which hardly preserve semantic distance between the original objects. Additionally, it is known that the usual Minkowski distances (e.g. Euclidean) used to compute similarity matrices do not behave well in high dimensional spaces, a phenomenon commonly referred to as the curse of dimensionality [26]. Therefore, in this paper, contrary to the usual approaches, we propose a fully categorical-like representation of a character configuration - which keeps our representation space's dimension relatively low - and apply adapted clustering methods to this transformed dataset.

The core idea behind the BaT approach is that each character configuration can be seen as a text document, composed of a list of keywords describing how the character is customized by the player. Here, inventory type elements, including both weapons and gears, come together with skill elements as well as specializations, to be merged into a single list. Numeric features are transformed into categorical bins (see Section III-B2).

The level of description for the keywords should be selected carefully, as it needs a good knowledge level of the core game mechanics. For example, in TCTD2 the skill variant is more important than the skill itself, because each skill variant has quite different gameplay mechanics; e.g. the fixer drone heals the player and stays close to him/her, while the assault drone pro-actively attacks distant opponents. In contrast, the weapon type (e.g. "Assault Rifle") is more important, in general, than the exact weapon model (e.g. "assault_rifle_pof416_t2_v1"), the latter variants mostly not influencing the underlying function in the build. Those choices in granularity level of the character configuration descriptors are driven by the concept of a build and are specific to the game. In this study, once base game mechanics have been well assimilated, the exercise of choosing the relevant level of granularity was intuitive and yielded comprehensive results.

Concretely, the list of keywords defining each character configuration is encoded using a classical text frequency data structure, and stored in a sparse matrix format, which makes it efficient for handling very large volumes of data.

2) Numerical features encoding into tiers: Some minor elements of a character configuration, taken individually, can have a lesser influence on the build compared to its main elements and are tedious to consider in a full equipment list. However, when combined over all possible available slots in an inventory, they can significantly impact some of the player's attributes. For example, *gear mods* in TCTD2 can buff up to 3 stats but with relatively low bonus values (e.g. critical hit chance +2%, critical hit damage +3%, weapon handling +5%) and only one attribute among offensive, defensive or utility. Properly stacked over all the inventory, they start making a difference and accounting for build intentions.

Data-wise, treating each piece of these minor modifiers individually could clutter the main build characteristics into a high number of low-relevance descriptors. To avoid this issue, we capture the essence of the player's optimization intent, through measuring the *consequences* of equipping full mods combinations. This is performed by computing the total score in each of following three attributes: offense, defense, and utility, as an integer number having values between 0 and a few 10s. Then we apply a quantile-based binning to each of these numerical features. Values in the 0-75% quantile range are encoded as *low_medium_tier*, and the values above as *high_tier*. Finally, we only keep the high tier instances and discard all the others, because we are only interested in

salient characteristics of each character configuration. Indeed, the low-medium tier values are expected to be the norm among players, thus would be present in most data points and would not carry meaningful information to differentiate between builds.

3) Handling repetitions: It is common in Role Playing Games (RPGs) that equipping the character with several pieces of a same 'set' provides powerful bonuses as a result of their combination, which are clearly communicated to the player. Those granted bonuses participate in the build construction since they provide advantages to specific attributes, related to different playstyles. For example, in TCTD2, a substantial bonus to marksman rifle damage is provided when equipping 3 gear pieces of *brand_airaldi_holding*: it is highly probable that players pursuing a sniper-like build will accordingly look for these combination bonuses. Data-wise, we translate this into a meaningful way for the clustering algorithm, during the data encoding step. Specifically, we keep all repetitions of the same gear element at brand level, without specifying their inventory slot (e.g. mask, chest armor, or gloves). We also discard the specific item names, since they have no influence on obtaining said set bonuses. For a complete example of a character configuration, see Table II.

TABLE II
EXAMPLE OF AN ENCODED CHARACTER CONFIGURATION

specialization_sharpshooter,
weapontype_marksmanrifle, *weapontype_marksmanrifle*,
weapontalent_breadbasket, *weapontalent_allegro*,
weapontalent_everlasting, *weapontalent_optimized*,
weapontalent_ranger, *weapontalent_stop_drop_and_roll*,
weapontalent_talent_exotic_marksman_mk1,
weapontalent_talent_exotic_marksman_mk1b,
weapontalent_talent_exotic_marksman_mk1_holstered,
brand_airaldi_holding, *brand_airaldi_holding*,
brand_airaldi_holding, *brand_airaldi_holding*,
brand_alps_summit_armaments, *brand_wyvern_wear*,
skillvariant_chemlauncher_repair_cloud,
skillvariant_hive_revive,
geartalent_spark, *geartalent_vigilance*,
geartalent_composure, *geartalent_concussion*,
geartalent_hard_hitting, *geartalent_opportunistic*,
geartalent_opportunistic, *offense_tier_high*,

4) Algorithm: We use Latent Dirichlet Allocation (LDA) algorithm, which is a classical text clustering algorithm [10]. It is very suitable to our purpose, since the clusters it produces - also called *topics* - are provided as an ordered list of keywords found in the dataset. The order of keywords for a given topic is defined by their relative importance in said topic. This output is extremely convenient for interpretation, since any non-data-science-proficient person having a good knowledge level of the game can make sense of these results and interpret each ordered keywords list as a build.

Another advantage of this algorithm is that it produces *soft clusters* instead of hard partitions of the data. In other words, for each character configuration in the input dataset, we obtain a probability distribution over all discovered topics (in contrast to just being assigned a single topic). In our context, this is advantageous because it enables us to obtain the most representative character configuration for each cluster/build. Examining the most representative configuration helps analysts and game designers understand and interpret the build much better than when looking at centroids obtained by numerical clustering approaches, which may not represent valid character configurations.

Furthermore, although classical topic modelling approaches do not always create semantically meaningful topics, in this work we apply them to keywords data and not natural language. Application of LDA to this type of data is particularly relevant, due to its underlying assumptions of probabilistic distributions over words and topics, considered under the 'bag-of-words' paradigm.

Several implementations of LDA are available in the open source ecosystem, most of them having excellent performances in terms of computing time and memory. In this paper, we use Python Gensim library [27], but other ones (such as Spark MlLib) would yield similar results.

TABLE III
LDA ALGORITHM HYPER-PARAMETERS

Hyper-parameter	Value
alpha	'asymmetric'[1]
eta	'auto'
chunk_size	1000
passes	10
workers	12

[1] alpha='auto' not available when using Distributed=True.

Used values for hyper-parameters can be found in Table III. We use the distributed variant of LDA to take advantage of multi-core computers at our disposal, and control the random seed for more reproducible results.

5) Selection of the number of topics: We try several values for the desired number of topics, which has a major impact on the results. To guide this exploration, we use topic coherence [28] as a measure of the quality of the resulting clustering for a fixed number of topics. To do this, we run the LDA algorithm over a range of values for this hyper-parameter, then keep the top 2 or 3 peak values of topic coherence as candidates for results analysis. This method considerably alleviates the usual issues with selecting the number of clusters, while keeping some flexibility concerning its final choice (See the results in Section IV-B1).

6) Entropy Filtering: For statistical purposes, we allocate each data point to a single cluster. To do this, we choose the the most probable build according to LDA results (see Section III-B4). But before this, for each data point we compute the normalized entropy of the build distribution (as determined by LDA) and apply a globally fixed threshold. Character configurations having a normalized entropy value higher than this threshold are considered having a too uncertain allocation, thus are assigned to an additional "Undefined" cluster.

IV. Results and Discussion

A. Numerical clustering approaches

1) Centroid-based clustering: Our initial approach was to encode a character configuration as a numerical vector and apply standard clustering algorithms that work in numerical feature spaces. Due to the many categories present in the dataset (different skills, weapons, armor etc.), this operation resulted in a representation that was mostly binary. As such, it was unsuitable for space partitioning techniques such as k-means as any results obtained bore little meaning.

2) Density-based clustering: As a second approach, we applied a density-based clustering method. Although we were able to re-discover known builds (see Section IV-B5), this was only possible after we had significantly biased the dataset by selecting features we assumed important for those known builds. This approach, however, only reinforces what we already know about character builds and might preclude our ability to discover new, unknown builds. Overall, we concluded that density-based clustering was not a suitable approach for build identification in TCTD2.

B. BaT Approach

In this section we present and discuss the results obtained by applying the BaT methodology described in Section III-B to the dataset described in Section II-C.

1) Selection of the number of topics: As described in Section III-B4, we ran the LDA algorithm with varying number of topics between 1 and 20, and computed the C_v coherence [28] for each resulting model. In order to obtain more accurate estimates, for each fixed number of topics we ran modelling in triplicates and computed corresponding C_v coherence mean and standard deviation. The results are shown in Fig.1. The whole process of the 3 x 20 LDA runs took no more than 8644s on a HP Z4 G4 Workstation (12 core Intel Xeon CPU @ 3.7GHz, 65 GB RAM).

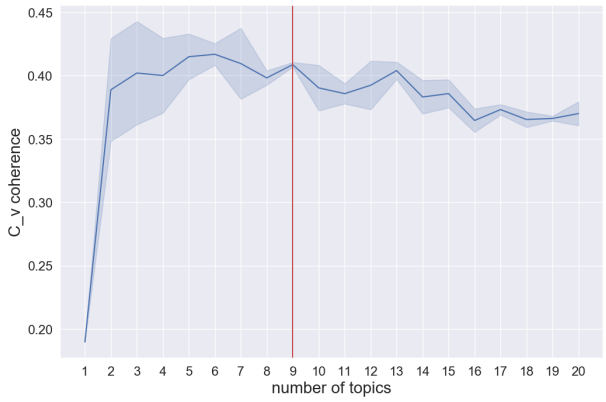

Fig. 1. C_v coherence as a function of the number of topics. Confidence intervals show +/-1 standard deviation based on triplicate estimates.

Fig.1 shows that the global average maximum value for C_v coherence (0.417) was reached for 6 topics. However, the representative keywords of these clusters were interpreted by game experts as amalgamations of several builds, indicating that a higher value for this hyper parameter should be preferred. Moreover, from their perspective, values between 8 and 12 were acceptable for a number of major builds in the game's "meta". Fig.1 clearly shows the appropriate value to be 9, which we subsequently used for the rest of the analysis.

2) Entropy Filtering: We applied the method described in Section III-B6, and used a threshold on normalized entropy of 0.5, which led to labelling only 10% of all character configurations as "Undefined".

3) Results for the chosen number of topics: The clusters found for 9 topics are listed in Table IV. For each of them, the characterizing keywords are listed in decreasing order of relative importance. To keep verbosity low, we limited the information shown in this table to the top 5 keywords, but the game experts used up to top 15 keywords to interpret the clusters. They attributed names to each cluster based on this input and their knowledge of the game. Later on, this task was said to be surprisingly easy, compared to previous attempts to make sense of high level in-game data aggregations.

An exhaustive description of the 9 discovered builds would be tedious, since their specificities are more related to the game itself, rather than the method. To illustrate the proposed methodology's ability to produce easily interpretable clusters, we hereafter detail two of them.

Build ID 0 is characterized by standard or *exotic* (a class of rare and unique items in TCTD2) sub-machine guns ("SMG"), together with *brand_sokolov* gear and *gearset_negotiators_dilemma*, both bringing high bonuses to SMG damage, critical hit chance, and critical hit damage. This combination is very powerful, and commonly known as the **SMG Sokolov** build.

Build ID 2 is characterized by several armor elements of *gearset_aces_and_eights* and *brand_airaldi_holding*, giving bonuses to damage done with long-range weapons and headshots. In this build, the main weapons are a standard marksman rifle, or the 'Nemesis' (a powerful exotic marksman rifle that provides bonuses for long-distance and focused shots). Moreover, this build mostly uses the 'sharpshooter' specialization, and it shows a high offensive tier, well fitting a "glass cannon" playstyle. This is the **Sniper** build.

Next, we visualized how the discovered builds relate to one another in terms of similarity and size. For this purpose, we applied a Principal Component Analysis to the distribution of keywords for each build, and plotted the projection onto the first two principal components (see Fig.2). Although it is only an approximation, it provides a useful overview of the relative sizes and positions of clusters in build space. This visualization also tends to confirm the results' relevance, since neighbor builds 0,4,7,8 share conceptual elements of playstyle, while distant builds (e.g. 2) are indeed distinct in terms of use of gear, weapons, and other attributes.

4) Clusters - Summary and Statistics: In order to better understand the discovered builds and study behavioral differences between them, we computed in a post-modelling phase various statistics and playtime metrics.

TABLE IV
CLUSTERS FOUND FOR 9 TOPICS - ORDERED KEYWORDS LISTS

ID	Build Name	Keywords
0	**SMG Sokolov**	*weapontype_smg*, *weapontalent_exotic_chatterbox*, *brand_sokolov*, *gearset_negotiators_dilemma*, *specialization_demolitionist*
1	**Seeker Mine**	*gearset_hard_wired*, *utility_tier_high*, *brand_alps_summit_armaments*, *skill_variant_seekermine_cluster*, *geartalent_exotic_btsu*
2	**Sniper**	*gearset_aces_and_eights*, *brand_airaldi_holding*, *weapontype_marksman_rifle*, *weapontalent_exotic_nemesis*, *specialization_sharpshooter*, *offense_tier_high*
3	**Clutch**	*brand_douglas_and_harding*, *brand_fenris_group_ab*, *weapontype_assaultrifle*, *geartalent_spark*, *offense_tier_high*
4	**Easy Exotic**	*brand_overlord_armaments*, *weapontype_rifle*, *weapontalent_exotic_lmg_pestilence*, *weapontalent_exotic_rifle_diamondback*, *weapontalent_exotic_lmg*
5	**Gunslinger**	*weapontalent_exotic_pistol_liberty*, *geartalent_exotic_holster_gunslinger*, *weapontype_lmg*, *weapontalent_unhinged*, *brand_petrov*
6	**Standard Raid**	*gearset_true_patriot*, *gearset_ongoing_directive*, *weapontype_assaultrifle*, *weapontype_lmg*, *offense_tier_high*
7	**Unbreakable**	*brand_gila_guard*, *brand_fenris_group_ab*, *defense_tier_high*, *brand_badger_tuff*, *geartalent_patience*
8	**Casual Joe**	*brand_providence*, *weapontype_assault_rifle*, *weapontype_lmg*, *brand_alps_summit_armaments*, *skill_variant_turret_assault*

TABLE V
CLUSTER SIZES

ID	Size	Dataset percentage
-1	23423	9.9%
0	9336	3.9%
1	37491	15.8%
2	9815	4.1%
3	26996	11.4%
4	3472	1.5%
5	3652	1.5%
6	16163	6.8%
7	35733	15.1%
8	70687	30.0%

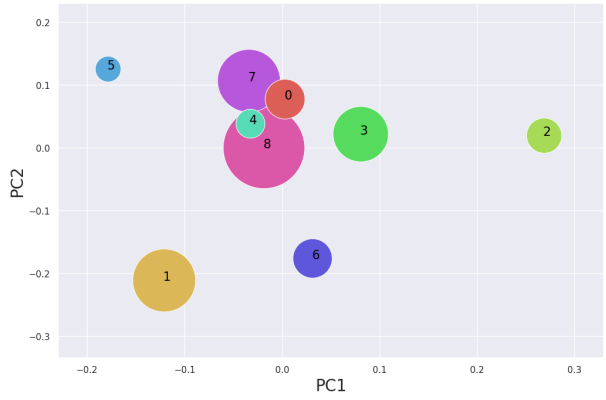

Fig. 2. 2D PCA projection of the nine-topic builds, labelled by build ID

Note: In tables V and VI, the cluster ID '−1' corresponds to the "Undefined" group, as described in Section III-B6.

Table V shows the number of character configurations allocated to each discovered build (i.e. the Size), following the method detailed in Section IV-B1, as well as the fraction of the dataset it represents. From this table we can, for example, see that build 8 is the most popular one among players, whereas builds 4 and 5 are rarely utilized.

Table VI shows the main, game-oriented statistics of each discovered build. The **Build attributes** columns of this table show the means and standard errors of the 6 considered attributes. The **Playtime split** columns show how players of each build spend their gaming time: firstly, in terms of the average fraction of time spent in a group vs. solo; And secondly, in terms of the fraction of time spent in main missions, dark zone (DZ), and other activities (exploration, side-missions etc.). The last three columns sum up to 100%.

Group playtime is usually utilized to indicate the tendency for social play [3]. We could argue that each build's playstyle comes with a different level of interdependence with other builds, consequently making it more or less suitable for increased social play. We can see for example that the "Sniper" build 2 shows a higher proportion of playtime in group than any other build. It would fit the context of the game that the less damage-resistant role of a sniper with a very high damage output (see Table VI) would need a team to get less exposed to close-quarter combat situations. More generally, the statistics obtained for the builds are in accordance to their function, gameplay style, and intended design.

5) Comparison with rules-based build assignment: To assess the relevancy of the proposed method, and as a sanity check, we tested whether BaT was able to recover at least a subset of builds expected to be present from external sources and game experts. Independently from BaT experiments, we performed a simple rules-based build assignment on the same dataset and compared the obtained results.

To do so, we manually defined 4 builds, well-known from online forums and internal playtests. Each of them could be assigned to a character configuration if and only if said data point satisfied all of the predefined build's conditions. Therefore, each data point could be allocated to *at most one* predefined build. When it was not allocated to any of them, the character configuration was labeled as "Undefined".

The 4 predefined builds were **Sniper**, **Clutch**, **Unbreakable**, and **Seeker**. More than 50% of character configurations could not be assigned to any build in this manner (see Table VII). We then closely inspected the 9-cluster results and identified 4 builds that most resembled the 4 predefined ones, based on their top descriptors (e.g. build with ID 7 from BaT most resembles the Unbreakable build from manual assignment etc.). The builds from the two clustering approaches were then paired up accordingly, and a Jaccard similarity index was

TABLE VI
CLUSTER SUMMARY STATISTICS

ID	Build attributes						Playtime split			
	Health score	Armor score	Skill power	Offense attribute	Defense attribute	Utility attribute	Group play (%)	Main missions (%)	DZ (%)	Other activity (%)
-1	77.5±0.3 K	253.7±0.4 K	1240±8	7.57±0.02	6.74±0.02	4.78±0.02	41.7±0.2	40.9	2.4	56.7
0	87.3±0.5 K	241.2±0.5 K	858±10	8.15±0.03	6.59±0.03	3.70±0.03	41.2±0.4	39.3	2.4	58.3
1	61.3±0.2 K	228.4±0.2 K	3207±5	4.34±0.01	4.99±0.01	11.60±0.01	47.9±0.2	44.0	2.4	53.5
2	62.4±0.3 K	223.2±0.4 K	291±5	13.63±0.04	3.92±0.02	1.35±0.02	51.7±0.4	37.3	3.2	59.5
3	109.7±0.4 K	218.7±0.3 K	339±4	11.53±0.01	4.33±0.01	1.65±0.01	48.5±0.2	33.6	6.8	59.6
4	77.1±0.7 K	262.7±0.9 K	868±17	8.04±0.05	6.95±0.05	3.39±0.04	40.8±0.7	40.1	2.0	57.9
5	66.1±0.5 K	260.8±0.8 K	1000±16	8.39±0.04	6.80±0.04	3.98±0.04	41.1±0.6	41.0	1.5	57.5
6	65.1±0.2 K	227.5±0.3 K	1165±8	9.58±0.03	5.32±0.02	5.37±0.02	39.5±0.3	43.4	1.0	55.6
7	94.5±0.2 K	295.0±0.3 K	669±4	6.63±0.01	9.73±0.01	2.88±0.01	41.9±0.2	35.0	4.1	60.9
8	69.8±0.1 K	223.3±0.1 K	1115±4	6.80±0.01	5.43±0.01	4.49±0.01	36.7±0.1	42.1	0.9	57.0

The build attributes section shows the mean value of each character attribute within a build cluster and its associated standard error of the mean. Playtime split shows the mean fraction of playtime spent in a group within a build cluster and its associated standard error of the mean. This is supplemented by the fraction of activity spent playing main missions, dark zone (DZ), and other in-game activities. The last three sum up to 100%

computed for each of the 4 pairs in order to get an estimate of the degree of assignment overlap. The results are presented in Table VII. The relatively high overlap scores (especially for the **Sniper** build: 45%) indicate that BaT was able to recover known builds to a large extent.

TABLE VII
COMPARISON OF BAT CLUSTERING AND RULES-BASED BUILDS

Build Name (Predefined)	BaT - frequency (%)	Predefined - frequency (%)	Jaccard index (%)
Seeker	16	12	16
Unbreakable	15	13	23
Clutch	11	18	35
Sniper	4	9	45

We also quantified the overall match between the rules-based clustering and BaT clustering results, without explicitly defining build pairs as in the previous paragraph. For this we used the adjusted random index (ARI) [29] and the adjusted mutual information (AMI) [30] indicators. ARI and AMI can be used to compare the degree of match between two clustering results even when the number of clusters produced by the two methods are not the same, which is the case in this study. The obtained values of 0.21 and 0.16 for ARI and AMI respectively, indicate a match that is better than random, but not perfect (a score of 1.0 would indicate a perfect match). This suggests that the BaT approach is capable of recovering known builds to a good extent, thereby passing the sanity check.

V. CONCLUSIONS

In this paper we dissected a case study in clustering inherently categorical game data highly dependent on the context and demonstrated a step-by-step method that encompasses the full process and application of the proposed method, from data collection and preparation to performance evaluation in context and against external data (e.g. socially emergent builds from game forums). We showed that the BaT method is more suitable than k-means or DBSCAN in producing comprehensive clusters. Our description of player behavioral data reflected by playtime split and character attributes showed that meaningful clustering of character builds results in representation of different playstyles and behavioral profiles that forms around their affordances. For data scientists, detailed illustration of our method may help in replication, performance improvement and incentive to experiment with less explored methods. Determination of significant behavioral differences between clusters also implies to game designers and scholars to study emergent character configurations beyond player performance metrics and encourage varied and personalized playstyles by monitoring player choices, behaviors incorporated with them and what they imply for player evolution. Furthermore, we can imagine this method being applied to other RPG games that allow gathering similar type of data, in order to better understand how players customize their characters.

VI. LIMITATIONS AND FUTURE WORK

There are some limitations to this work, which open several leads to improve and continue it. First, during the data collection step we kept only the last updated character configuration by each player over the studied period. This might not be the most relevant instance of a player build. Choosing alternative ways to snapshot character configurations is a challenging task as players tend to continuously modify them, whether for pure exploration (theory-crafting) or regular improvements (better gear found, levelling up, etc.). Finding better ways to detect relative stability in character configurations, and to study builds evolution in time and per player deserves a more rigorous research work. Alternatively, monitoring the evolution of successive clusterings after different title updates could also lead to interesting insights concerning the game's meta evolution, i.e. how players adapt their builds to in-game mechanics changes, new items/skills, and existing items modifications for balancing.

Second, the categorical nature of the data makes the proposed method quite sensitive to the artefacts that can appear during data pre-processing. Beyond the pure necessities of carefully crafting the data cleaning step, some very specific

and complex game mechanics can somehow clutter the obtained clusters. This was the case in this study focused on TCTD2, when dealing with a special class of end-game items (e.g. the *Exotics* and *Gear sets*) that have unique properties not found in usual gear. Developing adapted encoding steps for these end-game specific items could lead to improved results. Another limitation related to data encoding is that the keywords in the learning dataset all have equal weight, which is not necessarily the best option because of complex interactions between different categories of character features. In TCTD2, this is the case with some gear talents, that are activated if and only if very specific conditions on attributes are satisfied which the current encoding does not take into account and could be a powerful driver or constraint for builds crafting. Similarly, the presence of very common items among the player base can skew the results. More research in this area could lead to some improvements.

Another topic of interest is the design of accurate player in-game performance indicators, not only based on completion speed of main missions. Correlating such KPIs with builds information would provide insights about the builds' respective strength, and that could also be followed over time and per individual player. This would, in turn, provide relevant information to game designers concerning the practical builds efficiency, and potentially help them to perform early and targeted game balancing adjustments.

Finally, the method still relies on human interpretation to give meaning to the obtained clusters. Although eased because of their characterization with simple keywords, this task is still prone to potential errors and interpretation bias. Moreover, we lacked game experts' time to perform a detailed analysis of the character configurations that were un-categorized because of too high normalized entropy. This could help validating the hypothesis that these data points correspond to hybrid builds.

ACKNOWLEDGMENT

The authors would like to thank Christoph Gansler and Johan Lindholm for their expert knowledge of TCTD2.

REFERENCES

[1] N. Shaker, S. Asteriadis, G. N. Yannakakis and K. Karpouzis, "Fusing Visual and Behavioral Cues for Modeling User Experience in Games", in IEEE Transactions on Cybernetics, vol. 43, no. 6, pp. 1519-1531, Dec. 2013.

[2] A. Saas, A. Guitart and Á. Periáñez, "Discovering playing patterns: Time series clustering of free-to-play game data", 2016 IEEE Conference on Computational Intelligence and Games (CIG), Santorini, 2016, pp. 1-8.

[3] A. Canossa, A. Azadvar, C. Harteveld, A. Drachen, and S. Deterding, "Influencers in Multiplayer Online Shooters", Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI 2019.

[4] A. Canossa, S. Makarovych, J. Togelius, and A. Drachen, "Like a DNA String: Sequence-Based Player Profiling in Tom Clancy's The Division", Proceedings of the Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2018), pp. 152–158, 2018.

[5] C. Holmgård, M. C. Green, A. Liapis and J. Togelius, "Automated Playtesting With Procedural Personas Through MCTS With Evolved Heuristics", in IEEE Transactions on Games, vol. 11, no. 4, pp. 352-362, Dec. 2019.

[6] Á. Periáñez, A. Saas, A. Guitart and C. Magne, "Churn Prediction in Mobile Social Games: Towards a Complete Assessment Using Survival Ensembles", 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 564-573.

[7] A. Drachen, C. Thurau, R. Sifa, and C. Bauckhage, "A Comparison of Methods for Player Clustering via Behavioral Telemetry", Proceedings of Foundations of Digital Games, 2013.

[8] P. Braun, A. Cuzzocrea, T. D. Keding, C. K. Leung, A. G. Padzor, and D. Sayson, "Game Data Mining: Clustering and Visualization of Online Game Data in Cyber-Physical Worlds", Procedia Computer Science, vol. 112, pp. 2259–2268, 2017.

[9] A. Azadvar and A. Canossa, "UPEQ: ubisoft perceived experience questionnaire," Proceedings of the 13th International Conference on the Foundations of Digital Games", FDG 18, 2018.

[10] D. M. Blei, A. Y. Ng and M. I. Jordan, "Latent Dirichlet Allocation", Journal of Machine Learning Research, vol. 3, 2002.

[11] C. E. Shannon, "A Mathematical Theory of Communication", Bell system technical journal, vol. 27, no. 3, pp. 379–423, 1948.

[12] G. N. Yannakakis and J. Togelius, "Modeling Players", Artificial Intelligence and Games, pp. 203–255, 2018.

[13] J. H. Kim, D. V. Gunn, E. Schuh, B. C. Phillips, R. J. Pagulayan, and D. Wixon, "Tracking real-time user experience (true): A comprehensive instrumentation solution for complex systems", CHI, 2008.

[14] G. N. Yannakakis and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction", IEEE Trans. on Computational Intl. and AI in Games, 1(2):121-133, 2009.

[15] T. Fields and B. Cotton, "Social Game Design: Monetization Methods and Mechanics", MK Pub., 2011.

[16] M. Etheredge, R. Lopes, and R. Bidarra, "A Generic Method for Classification of Player Behavior", Ninth Artificial Intelligence and Interactive Digital Entertainment Conference, 2013.

[17] Y. G. Li, "A Clustering Method Based on K-Means Algorithm", Applied Mechanics and Materials, vol. 380-384, pp. 1697–1700, 2013.

[18] K. J. Shim and J. Srivastava, "Behavioral profiles of character types in EverQuest II", Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games, 2010.

[19] A. Drachen, R. Sifa, C. Bauckhage, and C. Thurau, "Guns, swords and data: Clustering of player behavior in computer games in the wild", IEEE Conference on Computational Intelligence and Games (CIG), 2012.

[20] O. Missura and T. Gärtner, "Player modeling for intelligent difficulty adjustment", In Proc. of the ECML-09 Workshop From Local Patterns to Global Models, 2009.

[21] F. Baumann, D. Emmert, H. Baumgartl, and R. Buettner, "Hardcore Gamer Profiling: Results from an unsupervised learning approach to playing behavior on the Steam platform", Procedia Computer Science, vol. 126, pp. 1289–1297, 2018.

[22] S.-J. Kang, Y. B. Kim, T. Park, and C.-H. Kim, "Automatic player behavior analysis system using trajectory data in a massive multiplayer online game", Multimedia Tools and Applications, vol. 66, no. 3, pp. 383–404, 2012.

[23] P. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis", J. Comput. Appl. Math., vol. 20, no. 1, pp. 53–65, 1987.

[24] J. C. Gower, "A General Coefficient of Similarity and Some of Its Properties", Biometrics, vol. 27, no. 4, pp. 857–871, 1971.

[25] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise", in Proc. of 2nd International Conference on Knowledge Discovery and, 1996, pp. 226–231.

[26] R. Bellman, "Dynamic programming", Princeton University Press, 1984.

[27] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora", Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, pp. 45–50, ELRA, May 2010.

[28] M. Röder, A. Both, and A. Hinneburg, "Exploring the Space of Topic Coherence Measures", Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM 15, 2015.

[29] L. Hubert and P. Arabie, "Comparing partitions", J. of Classification, vol. 2, no. 1, pp. 193–218, 1985.

[30] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?", in ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, Quebec, Canada, 2009, pp. 1073–1080.

[31] C. Cook, R. Conijn, J. Schaafsma, and M. Antheunis, "For Whom the Gamer Trolls: A Study of Trolling Interactions in the Online Gaming Context," Journal of Computer-Mediated Communication, 2019.

[32] G. Wallner, S. Kriglstein, and A. Drachen, "Tweeting your Destiny: Profiling Users in the Twitter Landscape around an Online Game," 2019 IEEE Conference on Games (CoG), 2019.