Retention Prediction in Sandbox Games with Bipartite Tensor Factorization

Rafet Sifa[1], Michael Fedell[2], Nathan Franklin[2], Diego Klabjan[2], Shiva Ram[2], Arpan Venugopal[2], Simon Demediuk[3], and Anders Drachen[3]

[1] Fraunhofer IAIS, Germany,
[2] Northwestern University, United States
[3] DC Labs, United Kingdom

Abstract. Open world video games are designed to offer free-roaming virtual environments and agency to the players, providing a substantial degree of freedom to play the games in the way the individual player prefers. Open world games are typically either persistent, or for single-player versions semi-persistent, meaning that they can be played for long periods of time and generate substantial volumes and variety of user telemetry. Combined, these factors can make it challenging to develop insights about player behavior to inform design and live operations in open world games. Predicting the behavior of players is an important analytical tool for understanding how a game is being played and understand why players depart (churn). In this paper, we discuss a novel method of learning compressed temporal and behavioral features to predict players that are likely to churn or to continue engaging with the game. We have adopted the Relaxed Tensor Dual DEDICOM (RTDD) algorithm for bipartite tensor factorization of temporal and behavioral data, allowing for automatic representation learning and dimensionality reduction.

Keywords: Tensor Factorization, Behavioral Analytics, Business Intelligence

1 Introduction

Game Analytics research has in recent years advanced rapidly. In the span of a decade, analytics has moved from a supporting role to a cornerstone of game development. Despite the commercial and academic interest, the domain is still in its explorative phase, with maturity of the knowledge, technology and models applied varying across business models, game genres and platforms [6, 5, 17].

Two key challenges in Game Analytics are player profiling and churn prediction. These are important for different reasons: *Behavioral profiling* is an important process in game development as it allows the complexity space of player behavior to be condensed into a specific set of profiles, which showcase how a game is being played. Behavioral profiling is notably important for persistent and semi-persistent games, where live operations utilize profiling to understand how the community is playing the game [17, 19]. *Churn prediction* is a key process in Game Analytics for many different types of games, not the least those that use a freemium revenue model [4, 8, 10, 16, 20]. Churn prediction basically attempts to predict when a specific player will stop playing

the game. With accurate churn prediction, it is possible for analytics teams to pinpoint players who may not be enjoying the game or experiencing problems progressing. Understanding when a player might leave the game provides the ability to explore why that might be happening via behavioral analysis [4, 8, 15, 16]. Churn prediction in open-world games, whether single-player or massively multi-player online (MMOG) is virtually unexplored, with very few publications on this problem [3, 21].

1.1 Objective and Contribution

While methodologically there are different possible approaches towards building profiles and classification models (see e.g. [19, 17, 16]), the approach adopted here is bipartite tensor factorization, due to prior successful application of tensor models in OWGs [21] and freemium games [22]. This paper is, to the best knowledge of the authors, the first to propose the use of bipartite tensor factorization for learning temporal representations for behavior prediction in games. The work presented directly extend prior Game Analytics research churn prediction, by showing incorporating low dimensional and automatically extracted temporal features can provide similar and for some metrics better prediction performance than models trained solely on aggregate behavioral data (that the majority of the previous work adopts) omitting the temporal information.

The test case used here is the open world game (OWG) *Just Cause 2* (JC2). JC2 features a massive freely navigable environment with missions, objectives and other activities spread across the environment. While AAA (major commercial) OWG titles like JC2 vary in their design (e.g. *Skyrim*, *Grand Theft Auto*, in general these feature spatio-temporal navigation, and tactical combat. Freedom is a characteristic of OWGs, and space/time are both important dimensions for assessing the user experience, and thus for behavioral analysis [6, 19]. On a final note, while MMOGs are typically also OWGs, the presence of many players within the same virtual world, a compared to just one for JC2, mean that the analyses presented here may not translate directly to these types of games.

1.2 Related Work

The work presented here builds on previous research in Game Analytics on prediction modeling, behavioral profiling and spatio-temporal behavioral analytics (e.g. [3, 12, 19, 15, 4, 8, 13, 25].

With respect to prediction in games, previous work has primary targeted either predicting future behavior [15, 12, 3, 13] or sought to inform situations related to agent modeling in Game AI [28]. In terms of the former, the emphasis has been on persistent or semi-persistent games where live operations are important to the financial success of a title. A variety of machine learning-based approaches have been adopted, including pattern recognition, regression, decision trees [8], support vector machines [27], Hidden Markov Models [16, 3] and deep learning [15]. Runge et al. [16] and Hadiji et al. [8] benchmarked multiple methods in churn prediction.

Behavioral profiling in games has a substantial history, recently summarized by Sifa et al. [19], and will therefore not be covered in detail here. The key objective of profiling

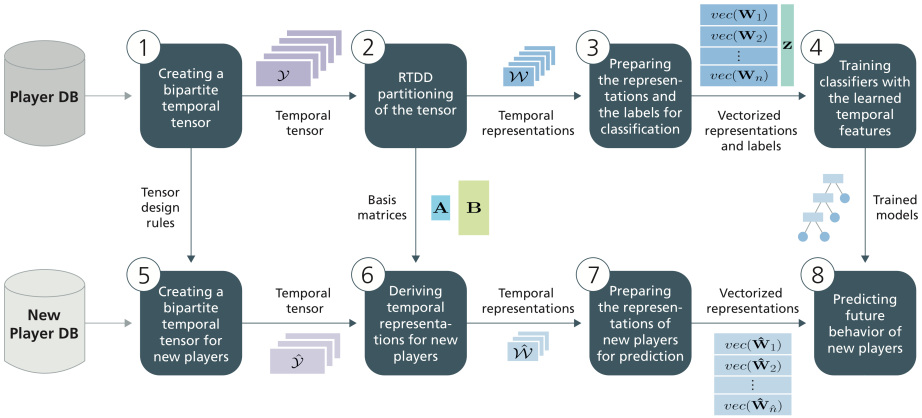


Fig. 1: Our bipartite tensor factorization based representation framework to compress multidimensional player information for future behavior prediction. Each slice of the extracted tensor encodes a player's observations and for training our predictors we consider the vectorized version (denoted by $vec(\cdot)$) of the low dimensional factors. In real life scenarios the learned basis matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be easily used to infer the low dimensional representations from new players (for orthogonal basis matrices this boils down to matrix multiplication), which can be fed to the trained classifiers for future behavior predictions.

is to act a means for managing complex user data and building meaning from them, discovering underlying patterns in the behavior of the players [18, 19]. Profiling allows for a condensation and modeling of a complex behavioral space, exemplified in MMOGs and OWGs. Spatio-temporal analytics is comparative infrequent, notably compared to the strong tradition in Game AI where e.g. agent models require consideration of both dimensions [28]. However, although several papers exist on the topic of visualizing behavioral data from games, e.g. Wallner et al. [26]. Another key precursor paper is Sifa et al. [21], who adapted different tensor models, that factorizes asymmetric waypoint matrices to learn spatio-temporal features, for churn prediction at the individual player level, achieving up to 81% accuracy for *Just Cause 2*, using the same dataset as applied here. The work by Sifa et al. [21] also highlighted the importance of spatio-temporal features in predicting retention in OWGs, possibly because these dimensions are integral to the user experience of these games. This result contrasted prediction work in mobile games where highly successful classification work has shown that spatial features are not important to predicting retention (see [25, 3, 12, 15, 4, 8]).

The work presented here extends [21] by considering a more general factorization model to be able to automatically learn temporal features from a set of bipartite player matrices. To this end, we will present how we can design a bipartite temporal behavioral player tensor and introduce the use of Relaxed Tensor Dual DEDICOM (RTDD) to extract features that can be later used in further analytics applications, which for our case will be about predicting the future arrival behavior of a set of JC2 players

2 Relaxed Tensor Dual DEDICOM

We devote this section to explain the tensor factorization model called Relaxed Tensor Dual DEDICOM (RTDD) model [11, 24, 22], which generalizes the matrix and tensor factorization models INDSCAL and DEDICOM [9, 21, 22] to decompose bipartite tensors into combinations of low ranked matrices. We will use this model in our experiments to factorize a data tensor that encodes temporal player interactions to learn compact player representations for retention prediction.

Formally, given a bipartite data tensor $\mathcal{Y} \in \mathbb{R}^{m \times n \times d}$ containing a collection d bipartite $m \times n$ matrices or *slices* (i.e. $\mathcal{Y} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_d\}$ for $\boldsymbol{Y}_i \in \mathbb{R}^{m \times n}$) and the dimensionalities of the hidden components p and q, the RTDD model yields a left basis matrix $\boldsymbol{A} \in \mathbb{R}^{m \times p}$, a right basis matrix $\boldsymbol{B} \in \mathbb{R}^{n \times q}$ and a coefficient tensor $\mathcal{W} \in \mathbb{R}^{p \times q \times d}$ to represent each slice $\boldsymbol{Y}_i$ of the data tensor as

$$\boldsymbol{Y}_i = \boldsymbol{A}\, \boldsymbol{W}_i\, \boldsymbol{B}^T, \tag{1}$$

where $\boldsymbol{W}_i \in \mathbb{R}^{p \times q}$ is the ith slice of $\mathcal{W}$.

It is worth mentioning that, akin to two factor matrix factorization models (see [17]), for a given set of factors $\{\boldsymbol{A}, \boldsymbol{B}, \mathcal{W}\}$ and the model parameters p and q, which are typically chosen to be $p, q \ll min(m, n)$, the representation in (1) compresses factorized tensor as the space complexities for the data and the factorized representation respectively are $O(mnd)$ and $O(mp + nq + dpq)$, where the latter reduces down to $O(dpq)$ when the coefficient tensors are used in further analytics applications (as we will show in our case study). Finding the RTDD factors of a given bipartite tensor $\mathcal{Y}$ can be obtained by minimizing the sum of the reconstruction error of each slice defined as

$$E(\boldsymbol{A}, \boldsymbol{B}, \mathcal{W}) = \sum_{i=1}^{d} \left\| \boldsymbol{Y}_i - \boldsymbol{A}\, \boldsymbol{W}_i\, \boldsymbol{B}^T \right\|^2, \tag{2}$$

where $\left\| \cdot \right\|$ is defined as the Frobenius norm [22], and cannot be directly solved due to the *unconvexity* of the factors in (2). A popular alternative to solve such problems is to consider a set of iterative optimization updates, in which objective function is optimized for each factor (for RTDD $\boldsymbol{A}$ $\boldsymbol{B}$ and each slice of $\mathcal{W}$) independently keeping the other factors fixed (see examples in [1, 11, 17]).

In summary, an alternating algorithm to come up with optimal RTDD factors starts with random factors[4] and iteratively optimizes the minimized objective E from (2) by consecutively

- minimizing E for $\boldsymbol{A}$ with fixed $\boldsymbol{B}$ and $\mathcal{W}$,
- minimizing E for $\boldsymbol{B}$ with fixed $\boldsymbol{A}$ and $\mathcal{W}$ as well as
- minimizing E for each slice of $\mathcal{W}$ with fixed $\boldsymbol{A}$ and $\boldsymbol{B}$

until a predefined stopping condition (e.g. a maximum number of iterations or stabilization of E) is met.

[4] The initial factors also has to follow the same constrains (if any imposed) for a converging optimization process.

(a) a look at the game world

(b) a fighting scene

Fig. 2: Just Cause 2 is an action-adventure sandbox (open world) game that takes place in an imaginary island Panau with a total coverage about 1000 square kilometers. Images are copyright of Square Enix (2009).

Another important aspect in tensor factorization is to impose constraints to the factors for efficiency in variety of aspects such as representability, interpretability or speed-up [17]. In this work we will consider the constraints introduced in [22][5] that force the basis matrices of RTDD $\boldsymbol{A}$ and $\boldsymbol{B}$ to be column orthonormal (i.e. $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_p$ and $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}_q$, where $\boldsymbol{I}_h$ is the $h \times h$ identity matrix). These constraints cannot only speedup the factorization process (as empirically shown in [23]) but also allow us to easily obtain coefficient matrices for a new set of players from the previously trained models. The latter is particularly beneficial in continuous profiling and prediction environments, in which behavioral representations are learned from a (typically) large player base and can be used to infer ones for newly observed data units.

For the case of RTDD, once a data tensor $\mathcal{Y}$ is decomposed into a combination of the factors $\{\boldsymbol{A}, \boldsymbol{B}, \mathcal{W}\}$ as in (1) we can obtain a coefficient tensor for an *unseen data tensor* $\hat{\mathcal{Y}} \in \mathbb{R}^{m \times n \times \hat{d}}$ (e.g. containing $\hat{d}$ players) by considering the global minimizers of (2) for each slice as $\hat{\boldsymbol{W}}_i \leftarrow \boldsymbol{A}^T\hat{\boldsymbol{Y}}_i\boldsymbol{B}$, where $\hat{\boldsymbol{W}}_i$ is the ith slice of the new coefficient tensor $\hat{\mathcal{W}}$ corresponding to the ith slice of $\hat{\mathcal{Y}}$ (see Fig. 1 for more details)..

3 Data and Pre-processing

In this section we will briefly explain the game, whose players we analyzed in this work, the important steps we considered for preprocessing our dataset and the way we designed our tensor for factorization with the goal predicting future player behavior.

3.1 Just Cause 2: Gameplay

Just Cause 2 is a third-person action-adventure game which allows players to explore an open world map, with the overarching goal of overthrowing the dictatorship government of the fictional nation of Panau (see Figure 2). The playable world is an area which covers about 1000 virtual square kilometers. The game allows players to use weapons

[5] We used the authors' original Python implementation from https://tinyurl.com/rtddcode in our experiments.

from a vast arsenal while giving them access to different kinds of sea, air, and land modes of transportation. To advance through the game, players can earn chaos points by completing missions or destroying select government properties, causing the government to collapse. The chaos system provides the players freedom to progress through the game in a number of different ways besides main mission completion.

3.2 Behavioral Features and Temporal Aggregations

The data set used in this analysis consists of in-game statistics of 5331 randomly sampled players. The dataset has more than 10 million records with actions, timestamps, and locations that were normalized and stored in a relational database for easier querying and processing. Based on an initial exploratory data analysis, nine unique players were removed from the dataset as they consisted of erroneous records with abnormal values of in-game statistics. The users are allowed to play the game at four different levels of difficulty. Since the dataset contains player statistics for each of the difficulty levels in which the player can engage the game; we considered a composite key of player id and difficulty level as a unique player. Thus, unlike the previous work analyzing this game, this analysis comprises of 6598 individual data points.

Similar to many of the previously mentioned early work in behavioral analytics in games, we extracted 93 features from our player base, which comprised our entire expanded behavioral space of interest. These features were recorded in the database as either cumulative statistics over the player's lifetime, or description of events that take place during gameplay. We note that these features can be categorized into four distinct groups. The largest group is comprised of the lifetime counter statistics of the game, which includes different kinds of kills, structures destroyed, chaos caused, missions completed, and many more; these values are recorded as lifetime totals at each increment. The next set of features is made of player actions not included in the statistics, such as entry and exit of vehicles and parachutes; these actions are both geo- and timestamped and are given as single, point-in-time observations. The third group of features pertains to the cause of deaths, and finally, the fourth group of features provides extraction (a form of transport) information.

To aggregate the behavioral features for each player, we required a common temporal feature space. Since the amount of time spent by a player for each session can vary considerably, we sought to design a temporal unit which would hold equivalence across players with minimal loss of information. To accomplish this, we divided the data by playtime (seconds played since starting) into Time Buckets, periods of 1000 seconds for the first $10,000$ seconds of game play, followed by periods of $50,000$ seconds until $1,000,0000$ seconds in total.

3.3 Labeling for Retention Analysis

Previous studies covering churn and retention analysis usually defined the prediction setting as observing the player within a predefined time interval and predicting his future behavior again in a predefined time interval (see [8, 21] for examples). Similar to [21] we also define a churning player as one who after an observation period of 14 days beginning with their first session, failed to return to the game in the 7 days following

Model	Precision	Recall	F-Score
Random Forest	0.605	0.657	0.631
Logistic Regression	0.450	0.675	0.541
Gradient Boosting	0.617	0.635	0.626

Table 1: Cross validation prediction performance of our baseline setting that omits the temporal features and only incorporates the behavioral features. We obtain up to 0.63 F-Score for predicting future player retention.

(days 15-21). Accordingly, we created a churn flag assigned to any player IDs who fit this definition. By this measure, roughly 30% of the unique players in our dataset were retained, and the other 70% churned. We note that since the number of churners is significantly high, it is of considerable benefit if we can rightly identify returners and churners using their gameplay data from the initial gaming sessions.

3.4 Final Tensor Design

After pre-processing the data, extracting behavioral features, and aggregating over the temporal units described above, the next step was to design the tensor for bipartite tensor factorization using the Relaxed Tensor Dual DEDICOM model which is the focus of this paper and will be subsequently discussed in greater detail. The processed dataset, had been aggregated to a long matrix format (or matricized) consisting of the records at the temporal unit (session/time bucket) for all players. Following that, the data were then scaled using the *standard* and *minmax* scaling. The former normalizes each feature to have 0 mean and standard deviation of 1, wheras, the latter normalizes every data feature to live in the same predefined range (we have chosen the most standard method to transform every feature to reside in the unit hypercube). This scaled long-format dataset was then converted back into a tensor with $m \times n \times d$ dimensions where d is the number of unique players, m is the number of temporal units, and n is a column for each behavioral feature.

Another important aspect of our tensor design was related to censoring. That is, the decomposition model requires each of it's d slices to have the same dimensions; however, the amount of time played by each player was widely varied. To remedy this, the value of m was chosen sufficiently large to capture the longest-playing player. All other player matrices were padded with rows of zeros for time units which exceeded their maximum playtime.

4 Prediction Results

In this section we will present our retention prediction results by first explaining the setting we considered for our baseline. Following that we will take a look at the prediction results using RTDD as input features from the perspectives of data normalization and parametrization.

		RTDD		Random Forest			Logistic Regression			Gradient Boosting		
p	q	Reconstruction Error	Iterations	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
25	15	1367.01	19	0.613	0.756	0.677	0.599	0.677	0.636	0.639	0.678	0.658
25	25	954.19	20	0.613	0.766	0.681	0.584	0.691	0.633	0.643	0.670	0.656
25	50	362.26	54	0.609	0.767	0.679	0.552	0.723	0.626	0.648	0.668	0.658
50	25	919.18	13	0.603	0.777	0.679	0.57	0.694	0.626	0.638	0.679	0.658
50	50	299.97	100	0.610	0.778	0.684	0.544	0.729	0.623	0.646	0.683	0.664

(a) prediction results incorporating minmax scaling for $\mathcal{Y}$

		RTDD		Random Forest			Logistic Regression			Gradient Boosting		
p	q	Reconstruction Error	Iterations	Precision	Recall	F-Score	Precision	Recall	F-Score	Precision	Recall	F-Score
25	5	71959.72	6	0.582	0.793	0.671	0.632	0.543	0.584	0.602	0.689	0.643
25	25	49984.64	13	0.579	0.797	0.671	0.568	0.667	0.613	0.629	0.625	0.627
50	5	72793.53	35	0.601	0.761	0.672	0.643	0.526	0.579	0.614	0.672	0.642
50	25	49714.19	100	0.593	0.791	0.678	0.568	0.672	0.616	0.621	0.646	0.634
50	50	21707.29	55	0.587	0.805	0.679	0.533	0.707	0.608	0.642	0.628	0.635

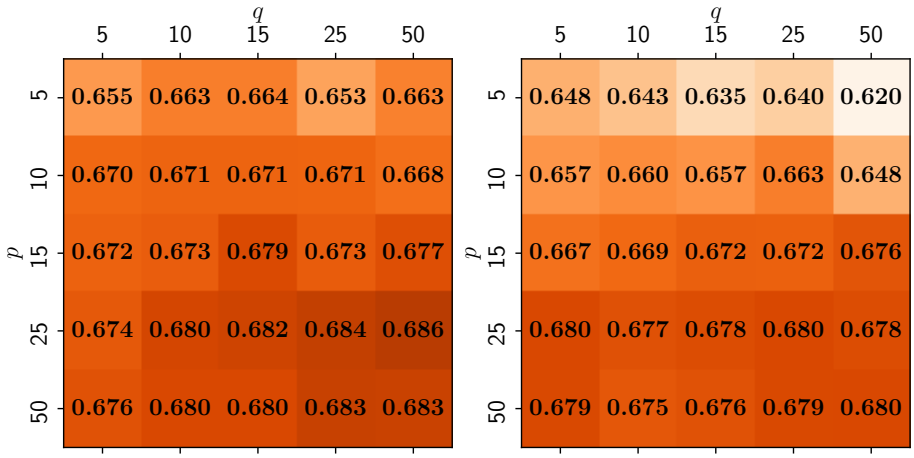
(b) prediction results incorporating standard scaling for $\mathcal{Y}$

Table 2: A more detailed comparison of the retention prediction results of our player-base for different parametrization of the tensor factorization model and normalization methods, where we obtained results that are better than our baselines (see Table 1).

In order to set a baseline, we analyze the performance of predicting retention using only the behavioral features ignoring the temporal axis. For this analysis we create a matrix with PlayerID as rows and the behavioral features aggregated over the 14 day activity period of a player to be the columns. We trained a 5-fold cross-validated Logistic Regression [14], Random Forest [2] and Gradient Boosting Classification [7] models with 93 aggregated behavioral features as predictors and retention flag as response to predict player retention. Among these models, Random Forest predicted retention the best with precision of 0.605, recall of 0.657 and F-Score of 0.63 (see Table 1). In the following we will use these results as our baseline.

We incorporated the temporal behavior of the players with their behavior over the 14 day period by creating a tensor $\mathcal{Y}$ as described above with time periods (temporal feature) as its rows, behavioral features as column and each individual players as slices. The behavioral features are aggregated across time periods from only 1-14 days in their playing life of each player. Following that we used RTDD to factorize tensor $\mathcal{Y}$ into *temporal basis matrix* $\boldsymbol{A}$, *behavioral basis matrix* $\boldsymbol{B}$ and coefficient tensor $\mathcal{W}$. RTDD embeds the temporal and behavioral dimensions into their respective loading matrices. Each slice of the coefficient tensor $\mathcal{W}$ is then *vectorized* and used as compact temporal-behavioral features to predict future user behavior. In order to evaluate the improvement in retention prediction brought about by the compact features, we predict retention probability of players using Random Forest, Logistic Regression and Gradient Boosting Classifiers.

As in our case the choice for the number of latent factors affects the representational power [17] and thus the follow-up applications using the latent representations,



(a) minmax scaling for $\mathcal{Y}$ (b) standard scaling for $\mathcal{Y}$

Fig. 3: Exploring the retention prediction quality in terms of F-Score for two popular scaling techniques and different parametrization of RTDD using the Random Forest classifier. For the former we chose the standard scaling, that normalizes the data to have zero mean and standard deviation one, and minmax scaling, that compresses all the features to a predefined range (usually to the unit hypercube). We ran a grid search for predicting retention in JC2 for RTDD parameters defined as $q, p \in [5, 10, 15, 25, 50]$. Our results indicate that, although for both of the utilized normalization methods the best results are obtained for larger values of p and q, compared to standard scaling, minmax normalization yielded more stable prediction results.

we utilized a grid search on the RTDD parameters p and q. We particularly chose the values of p and q to be respectively as $p \in [5, 10, 15, 25, 50]$ and $q \in [5, 10, 15, 25, 50]$, while assuring $p, q \ll min(m, n)$ to consider a compressed factor representation for each player. We trained Random Forest classifier with the compressed features as predictors and retention flag as response to predict retention probability of players. 5-fold cross validation was used to evaluate the Random Forest Model on various settings of tree depth and maximum features used to split the decision nodes in the trees while building the ensemble model. All the resulting models were evaluated based on their cross-validated F-score value. We present the prediction results for different values of p and q in Figure 3. We note that the models ran with compact features from minmax scaled tensors predicted retention slightly better than standard scaled tensors for the same p and q setting. After our explorative analysis, we then created a smaller subset of optimal p and q settings based on highest 5 settings with cross-validation F-score from the Random Forest output (that we show in Table 2). For p value of 50 and q value of 50, Random Forest predicted player retention best with precision, recall and F-score of 0.61, 0.778 and 0.684 respectively. Following that, to compare the Random Forest results against other standard classification models, we trained Logistic Regression and

Gradient Boosting classifier for the smaller subset of optimal p-q settings (see Table 2 for cross validation precision, recall and F-score comparisons).

Overall, our results do indeed indicate that, compared to the baseline model, the compact temporal behavioral features learned with RTDD have improved the retention prediction results substantially, where observed improvements more than 0.8%, 14% and 5% for respectively the values of precision, recall and F-score. This implies that bipartite tensor factorization not only allows for learning compact temporal representations but also informative representations that help us predict future behavior better than the non-temporal behavioral features.

5 Conclusion and Future Work

In this work we presented a novel approach that is based on the work of [21] to automatically learn useful representations from temporal and behavioral features in sandbox games by utilizing bipartite tensor factorization. Unlike the static feature definitions that are mostly utilized in the previous work (e.g. as in [8, 15]) our approach easily allows to incorporate the temporal player behavior into any prediction framework. Our case study with JC2 empirically showed that incorporating the coefficient matrices that are automatically learned by factorizing our tensor (encoding information about players, time periods and behavioral features) for each player can improve predicting the future arrivals. Our future work involves evaluating our behavior prediction models for the cases of enforcing different constraints (than orthogonality of the basis) on our tensor factorization model. In addition to that, we will explore how adding such constraints impacts the interpretability of the resulting factors.

Bibliography

[1] B.W. Bader, R. Harshman, and T.G. Kolda. Temporal Analysis of Semantic Graphs using ASALSAN. In *Proc. IEEE ICDM*, 2007.

[2] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[3] Simon Demediuk, Alexandra Murrin, David Bulger, Michael Hitchens, Anders Drachen, William L Raffe, and Marco Tamassia. Player Retention in League of Legends: A Study Using Survival Analysis. In *Proc. ACSW IE*. ACM, 2018.

[4] A. Drachen, E. Lunquist, Y. Kung, P. Rao, D. Klabjan, R. Sifa, and J. Runge. Rapid Prediction of Player Retention in Free-to-Play Mobile Games. In *Proc. AAAI AIIDE*, 2016.

[5] A. Drachen, P. Mirza-Babaei, and L. Nacke. *Games User Research*. Oxford University Press, 2018.

[6] El-Nasr, M. and Drachen, A. and Canossa, A. *Game Analytics: Maximizing the Value of Player Data*. Springer, 2013.

[7] Jerome H Friedman. Stochastic gradient boosting. *Computational statistics & data analysis*, 38(4):367–378, 2002.

[8] F. Hadiji, R. Sifa, A. Drachen, C. Thurau, K. Kersting, and C. Bauckhage. Predicting Player Churn in the Wild. In *Proc. IEEE CIG*, 2014.

[9] R. A. Harshman. Models for Analysis of Asymmetrical Relationships among N Objects or Stimuli. In *Proc. Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology*, 1978.

[10] K.-J. Kim, D. Yoon, J. Jeon, S.-I. Yang, S.-K. Lee, E. Lee, P. Bertens, A. Periez, F. Hadiji, M. Mller, Y. Joo, J. Lee, and I. Hwang. Game Data Mining Competition on Churn Prediction and Survival Analysis using Commercial Game Log Data. In *Computational Intelligence and Games (CIG)*, 2017.

[11] T. G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009.

[12] S. K. Lee, S. J. Hong, S. I. Yang, and H. Lee. Predicting Churn in Mobile Free-to-play Games. In *Proc. IEEE ICTC*, 2016.

[13] Xi Liu, Muhe Xie, Xidao Wen, Rui Chen, Yong Ge, Nick Duffield, and Na Wang. A semi-supervised and inductive embedding model for churn prediction of large-scale mobile games. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018.

[14] P. McCullagh and J.A. Nelder. *Generalized Linear Models, Second Edition*. Chapman and Hall/CRC Monographs on Statistics and Applied Probability Series. Chapman & Hall, 1989.

[15] Á. Periáñez, A. Saas, A. Guitart, and C. Magne. Churn Prediction in Mobile Social Games: Towards A Complete Assessment Using Survival Ensembles. In *Proc. IEEE DSAA*, 2016.

[16] J. Runge, P. Gao, F. Garcin, and B. Faltings. Churn Prediction for High-value Players in Casual Social Games. In *Proc. IEEE CIG*, 2014.

[17] R. Sifa. *Matrix and Tensor Factorization for Profiling Player Behavior*. LeanPub, 2019.

[18] R. Sifa and C. Bauckhage. Online k-maxoids clustering. In *2017 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 667–675. IEEE, 2017.

[19] R. Sifa, A. Drachen, and C. Bauckhage. *Profiling in Games: Understanding Behavior from Telemetry*. 2017.

[20] R. Sifa, F. Hadiji, J. Runge, A. Drachen, K. Kersting, and C. Bauckhage. Predicting purchase decisions in mobile free-to-play games. In *Eleventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2015.

[21] R. Sifa, S. Srikanth, A. Drachen, C. Ojeda, and C. Bauckhage. Predicting Retention in Sandbox Games with Tensor Factorization-based Representation Learning. In *Proc. IEEE CIG*, 2016.

[22] R. Sifa, R. Yawar, R. Ramamurthy, and C. Bauckhage. Matrix and Tensor Factorization based Game Content Recommender Systems: A Bottom-up Architecture and A Comparative Online Evaluation. In *Proc. AAAI AIIDE*, 2018.

[23] R. Sifa, R. Yawar, R. Ramamurthy, C. Bauckhage, and K. Kersting. Matrix- and Tensor Factorization for Game Content Recommendation. *Springer German Journal of Artificial Intelligence*, 2019.

[24] L. R Tucker. Some Mathematical Notes on Three-mode Factor Analysis. *Psychometrika*, 31(3):279–311, 1966.

[25] Markus Viljanen, Antti Airola, Tapio Pahikkala, and Jukka Heikkonen. Modelling User Retention in Mobile Games. In *Proc. IEEE CIG*, 2016.

[26] Günter Wallner and Simone Kriglstein. Plato: A visual analytics system for gameplay data. *Computers & Graphics*, 38:341–356, 2014.

[27] H. Xie, S. Devlin, D. Kudenko, and P. Cowling. Predicting Player Disengagement and First Purchase with Event-frequency Based Data Representation. In *Proceedings of Computational Intelligence in Games (CIG)*, 2015.

[28] Georgios N Yannakakis and Julian Togelius. A panorama of artificial and computational intelligence in games. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4):317–335, 2015.