Proceedings of the Fourteenth Artificial Intelligence and
Interactive Digital Entertainment Conference (AIIDE 2018)

Matrix and Tensor Factorization Based Game Content Recommender Systems: A Bottom-Up Architecture and a Comparative Online Evaluation

Rafet Sifa
Fraunhofer IAIS, Sankt Augustin, Germany
University of Bonn, Bonn, Germany
Rafet.Sifa@iais.fraunhofer.de

Raheel Yawar
Flying Sheep Studios, Cologne, Germany
RWTH Aachen, Aachen, Germany
Raheel.Yawar@rwth-aachen.de

Rajkumar Ramamurthy
Fraunhofer IAIS, Sankt Augustin, Germany
University of Bonn, Bonn, Germany
Rajkumar.Ramamurthy@iais.fraunhofer.de

Christian Bauckhage
Fraunhofer IAIS, Sankt Augustin, Germany
University of Bonn, Bonn, Germany
Christian.Bauckhage@iais.fraunhofer.de

Abstract

Players of digital games face numerous choices as to what kind of games to play and what kind of game content or in-game activities to opt for. Among these, game content plays an important role in keeping players engaged so as to increase revenues for the gaming industry. However, while nowadays a lot of game content is generated using procedural content generation, automatically determining the kind of content that suits players' skills still poses challenges to game developers. Addressing this challenge, we present matrix- and tensor factorization based game content recommender systems for recommending quests in a single player role-playing game. We discuss the theory behind latent factor models for recommender systems and derive an algorithm for tensor factorizations to decompose collections of bipartite matrices. Extensive online bucket type tests reveal that our novel recommender system retained more players and recommended more engaging quests than handcrafted content-based and previous collaborative filtering approaches.

Introduction

Recommender systems have become important tools of the trade in e-commerce, where they provide personalized suggestions to users who have to browse vast product portfolios (Smith and Linden 2017). Players of digital games, too, face numerous decisions, both in-game (e.g. choosing quests, characters, or tactics) and out-game (e.g. buying additional downloadable content packages in semi-persistent games, making In-App Purchases in free-to-play (F2P) games, or switching to different games on online gaming platforms) (Runge et al. 2014; Sifa, Bauckhage, and Drachen 2014; Ryan et al. 2015b; Saas, Guitart, and Perianez 2016). Such decisions impact their gameplay experience and result in player retention or churn. Game producers are therefore interested in building recommendation systems to provide tailored game content to their users in order to keep them engaged. In this work, we focus on recommender systems for game content, in particular, on recommending quests in an F2P hack and slash styled role-playing game known as *Trollhunters: Adventures in the Troll Caves*. In addition to the scripted main quests that contribute to the overall progression of the game, players are offered side quests which are pre-generated by procedural content generation. In order to keep the players engaged and increase player retention, the difficulty of these side quests should match with user's skills maintaining a feeling of flow (Chen 2007) i.e. the difficulty is neither so hard that the users give up nor so easy that they get bored from the lack of challenge. Several other factors contribute to the level of enjoyment such as the type of quests, quest duration, weapons available in each quest etc. However, identifying such features require a lot of analysis, given the number of players is large, such an approach is highly impractical. Therefore, we mainly focus on data-driven approaches such as learning representations from player-quest interactions and build a recommender system based on latent factor analysis.

Developing this in-game quest recommender system posed a number of challenges. Primarily, due to a large number of players and its inherent online nature, any such system has to be trained incrementally online. Secondly, the game itself was being developed along with the recommender system, bundled together and released to the users as a one-time deliverable. This means that the traditional process of building a recommender system by *build, test and tune* approach is not a viable option in our case. Also, the evaluation metrics used in an off-line batch setting is not well suited. Therefore, our contributions in this paper mainly focus on addressing these challenges in building a framework for such an online quest recommender system. First, we derive an extension of a tensor factorization method which could be trained iteratively. Next, to evaluate our models, we perform bucket testing of our models on different user groups and consider player retention as our evaluation metric since it implicitly measures player engagement. Finally, we compare three different engagement metrics of each group to further assess the quality of the recommendations.

Recommender Systems in Games

Grouping the previous work based on the type of input data used to generate the recommendations, we observe that most of the work on recommender systems has been built on contextual game related as well as in-game behavioral data.

Previous work in the former group includes contextual data in the form of text and the proposed recommendation approaches take advantages of vector space models when finding similar players that we will also consider for our content recommender systems. The work from (Meidl, Lytinen, and Raison 2014) applies information theoretic approach to co-cluster occurrences of adjectives and context words in user reviews to find similar games for recommendations. The proposed method represents each user review in a vector space that is defined by the frequencies of all the co-occurrences of its adjectives and context words and was evaluated off-line based on reviews from ten players. Following that, a matrix factorization based game recommender system in form of a search engine has been proposed by (Ryan et al. 2015b). Building on that work, (Ryan et al. 2015a) utilize matrix factorization to build a context based recommender system, that recommends games to players based on their game reviews. The authors evaluate their recommendation results based on surveying ten people and conclude that matrix factorization provides a significant increase compared to the baseline recommender in terms of matching accuracy.

Continuing with the studies related to using in-game behavioral data for building recommender systems, we note similarities to the former case in terms of utilizing vector space representations for performing recommendations, however, we observe more variety in terms of applications. In (Sifa, Bauckhage, and Drachen 2014) the authors proposed the use of a constrained two-way matrix factorization model to build a game recommender system based on playtime information, which (unlike any type of game rating) in the recommender systems literature is classified as a type of implicit feedback. The authors evaluated the generalization of their methods in an off-line fashion by predicting the playtime of holdout games. Similarly the industry case study from (Weber 2015) includes neighborhood based recommendation approaches that for each user rank in game items with respect to item inventory of other similar players. (Sifa et al. 2018) propose a hybrid profiling based playstyle recommender system using matrix factorization and evaluated their system in an online fashion by conducting a survey about the motivation of the analyzed thirty players.

Our work differs from the previous work in several respects. First, to the best of our knowledge, our work is the to first to publish about an architecture of a large scale recommender system for game content. Second, our work provides an online evaluation by bucket testing the performance of different recommendation algorithms and their settings in the context of game analytics. Third, our work is the first to build tensor factorization based recommender systems for collaborative filtering in games. To this end, we extend the algorithm from (Sifa et al. 2016) to factorize tensors of certain structural constraints and derive algorithm to decompose tensors that contain a collection of bipartite matrices.

Trollhunters: Adventures in the Troll Caves

We perform our evaluation via a video game. Its original title is: *Trolljäger: Abenteuer in den Trollhöhlen* [1]. It is an ad-driven free-to-play role-playing hack and slash dungeon crawler (see Fig. 1), where users take control of the protagonist, who is accompanied by two AI non-player characters (NPCs). The core loop of the game consists of the player starting in a dungeon marketplace, choosing a quest offered by two out of three NPCs, entering a procedurally generated dungeon, completing the quest, and finally returning to the marketplace. The quest phases can include finding gems, fighting enemies, and, occasionally, finding lost friends. Completion of each quest gives experience points which increase the player level. These levels can be invested in one of the three player attributes of strength (health), agility (movement and attack speed) and damage. The player character maxes out at level 30 and can invest only ten points in each attribute. Each quest has one to three phases. Quests are classified into two types: story quests and side-quests, that are randomly generated using a procedural content generation algorithm.

The generator for the latter takes the difficulty value, and seed as parameters and the difficulty value defines how strong the enemies will be. The players can complete a quest if they finish all phases. They can also fail it if they die or can abandon/quit it from the in-game menu if they dislike it. The procedural quest generator uses pre-built dungeon pieces which are put together to generate a complete level. It was used to create 85,000 quests. Using a client-server architecture, the game is rendered at the client-end, and the recommendation algorithms run at the server-end.

Following a tutorial quest, in order to familiarize the player with the game, build a user profile, and avoid the cold start problem, the user is given a completely random pool of quests by the server up until level 3. Following that we continue to send random quests in the case of our control group (Baseline), and all other groups receive quests from their respective recommendation algorithms.

A Content Based Recommender System

Our content-based (CB) quest recommendation approach is an intricate and iteratively developed rule-based solution inspired by (Leskovec, Rajaraman, and Ullman 2014) and divided into a profile learner and a filtering component. Since we generate our content procedurally, we do not need to categorize our quests as each quest's discrete information is already known. The primary focus of the profile learner is to find the most suitable quest for players based on their skill. Its secondary focus is to use players' preferences to find the most like-able quest. With our content-based method, we work with the assumption that the player's skill is not always deducible from the player's progress in the game.

To estimate a user's skill level based on the updated profile, we apply linear smoothing to Health Lost and Accuracy of all quests completed so far, giving the most recent ones a higher weight than older ones. Reason being, as the players learn the mechanics, their performance will improve. Similarly, a

[1]The game is an HTML5 application hosted on Toggo at the link: http://www.toggo.de/serien/trolljaeger/index-4310.htm

(a) interaction with a quest giver

(b) the quest dialogue

(c) a combat scene

Figure 1: The core loop of *Trollhunters: Adventures in the Troll Caves* (original: *Trolljäger: Abenteuer in den Trollhöhlen*) consists of choosing a quest offered by NPCs and entering the dungeon to complete all phases of the quest. Hack and slash styled combat is the primary game mechanic. (a) shows one of the three NPC quest givers where the exclamation point and circle indicate that a quest is available. (b) shows the quest dialogue for the quest phases (three in this case) and steps of each phase. (c) shows the protagonist and AI companion (in the back) in combat with two trolls (on the right).

player returning to the game after a hiatus might perform poorly, so we would like to reduce the difficulty before ramping it up again. For the total number of m quests completed by a user and a normalization coefficient for the jth quest $w_j = j \Big/ \sum_{i=1}^{m} i$ we define the weighted averages of Accuracy and Health Lost respectively as $a_u = \sum_{j=1}^{m} w_j a_j$ and $h_u = \sum_{j=1}^{m} w_j h_j$, where a_j and h_j are respectively the recorded accuracy and health loss values of quest q_j.

We then calculate the average and standard deviation of the Accuracy represented by A_a and A_s respectively across the complete set of quest results performed by all users of the group. We will use this as a baseline when computing the difficulty value of an individual user. We set the average Health Lost H_a to a heuristic value of 40% and calculated its standard deviation H_s from all user results. This heuristic complies with our previous discussion about flow (Chen 2007) i.e. the difficulty level should neither be too easy nor too hard. The consensus of using this value was reached during several rounds of playtesting. We apply linear smoothing to the failure rate r_u as well, but it takes on a slightly different form. Firstly, we take the ratio of quests failed at each level q_{f_l} versus the total quests attempted at each level q_l weighted in the order of the player level l when the quest was attempted, where L is the maximum level which is 30. Given our weighting coefficient $w_l = l \Big/ \sum_{i=1}^{L} i$ we define the failure rate as $r_u = \sum_{i=1}^{L} w_i (q_{f_i}/q_i)$. We also give the failure rate a heuristic value. In the pilot study, the average failure rate R_a was set at 10% with a standard deviation of R_s of 5% which resulted in a player failure rate of 25.94%. We reduced the average failure rate to 1% with the standard deviation of 0.5% which as presented in the evaluation section, reduced the actual player failure rate.

Following that, using the three mentioned measures, we estimate the player's difficulty value. Player levels range from 1 to 30, and our Baseline receives quests that have a difficulty equivalent to their level. In content-based approach, the difficulty, need not be equal to the player's level. We compute it by measuring how better or worse the player is performing compared to our global or heuristic averages for which we use piecewise linear functions. The difference or what we call, the variance between the player's Health Lost, Accuracy and Failure Rate values compared to the global or heuristic values, allows us to compute how much the player's preferred difficulty value might vary from the player's default difficulty, i.e. the player level. In the case of Health Lost and Accuracy, if the player's values are above the global or heuristic values, our difficulty variance is positive, and if they are below, our difficulty variance is negative. However, the Failure Rate only influences the difficulty variance if it drops below the heuristic value; otherwise, its variance is set to zero.

Once we have the three variance values, we take their weighted sum. The weights in this case are set based on how much they should impact a player's difficulty value. We set the weights for Accuracy, Health Lost and Failure Rate to 18.33%, 15% and 66.67% respectively. We note that the difficulty variance range is set in the interval $[-3, 3]$, rounded up to the nearest integer. The weights and the intervals both were computed during playtesting by iterating through several variations. The variance is added to or subtracted from the player's level to compute the new difficulty value. This value is then stored permanently as part of the player information and is recomputed each time the player finishes a quest. Whenever the player needs a fresh batch of quests to choose from, we recommend the quests that have the player's difficulty value instead of their level.

The secondary function of the content-based approach is to predict user preferred quests. A quest can take place in one of two environments, one of the three quest giver NPCs can offer it, and it can have one out of the three unique phases. To create the user's preference profile, we divide the number of quests played with the mentioned features, by the total number of quest played. This gives us probabilities of each choice of each of the three features. While recommending quests, we favor the quests with the features that the player prefers. One pitfall of this method and any other content-based approach, in general, is over-specialization (Leskovec, Rajaraman, and Ullman 2014) to avoid which, 30% of the quests we send to the user are chosen randomly.

Collaborative Filtering For Quest Recommendations

One of the fundamental problems of rule-based recommender systems such as the one we described above is the diversity in user behavior. This makes it very expensive to build and maintain new systems; as even slight changes in the recommendations usually require numerous iterations and extensive playtesting to find the right balance in the game and not lead to player churn. Data-driven methods, in particular, collaborative filtering algorithms, aim to circumvent these issues by automatically capturing different behavioral aspects and being straightforward to train to model inherently distinct user behavior (Smith and Linden 2017).

In the context of games, among numerous data-driven recommender systems techniques, neighborhood oriented collaborative methods have been widely adopted due to their easiness of implementation and success in practical applications (Sifa, Bauckhage, and Drachen 2014; Weber 2015; Ryan et al. 2015a). The main idea behind such methods is to recommend the (active) players new content-based on matching their content consumption history against the one from other similar players. Akin players, in this case, can be found by considering their similarity in the content space, which is the vector space defined by the analyzed content features. A commonly used similarity measure that is bounded and (unlike the euclidean distance) rather concentrates on tendencies than size is the *cosine* similarity, which for given two vectors $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^m$ can be realized by normalizing the dot product between two vectors by both their distances as $\sigma(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i^T \mathbf{v}_j}{||\mathbf{v}_i|| \, ||\mathbf{v}_j||}$. Example recommender systems that make use of player representations in vector spaces include the (weighted) bag of words when modeling game reviews in (Ryan et al. 2015a; Meidl, Lytinen, and Raison 2014) or playtime representations from (Sifa, Bauckhage, and Drachen 2014).

Considering the task of quest recommendation in our game, as we do not possess any explicit feedback to quantify player's satisfaction level we consider the implicit feedback by means of their behavioral activities for the played quests. Following that, unlike conventional single item-user representations as bipartite matrices, we consider a more generalized approach for our case and adopt a third-order tensor representation (Kolda and Bader 2009; Zook et al. 2012; Sifa et al. 2016), where we build our recommender systems by grouping different bipartite matrices containing certain behavioral aspects. That is, given n players, m quests and d different behavioral aspects that are related to the played quests, we consider a collection of matrices $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_d\}$, where each slice $\boldsymbol{X}_c \in \mathbb{R}^{m \times n} \; \forall \; 1 \leq c \leq d$ and x_{cji} represents the performance of the cth feature of ith player at the jth quest. For our case, we consider $d = 7$ behavioral aspects that include user level, strength, agility, damage, accuracy, health lost and completion time. Given that, we can build a neighborhood oriented (NO) recommender system by individually calculating the cosine similarity between two arbitrary players for each content matrix $\boldsymbol{X}_c$ and combine the results as a weighted sum over the slices by considering the composite similarity measure

$$\hat{\sigma}(\mathbf{X}_{::i}, \mathbf{X}_{::j}) = \sum_{c=1}^{d} w_c \sigma(\mathbf{x}_{c:i}, \mathbf{x}_{c:j}), \tag{1}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a stochastic (nonnegative) vector, $\sigma(\cdot, \cdot)$ represents the cosine similarity, $\mathbf{X}_{::i} \in \mathbb{R}^{m \times d}$ represents the matrix containing the ith columns of each slice and $\mathbf{x}_{c:i} \in \mathbb{R}^m$ represents the ith column of the cth slice of $\boldsymbol{\mathcal{X}}$.

Yet, one particular characteristic of neighborhood-oriented methods is that they naturally favor assigning high similarity values to players that have only played the same games together and might not produce recommendations for similar quests. To circumvent that, we can use matrix factorization methods, that can reveal the hidden structures in the quest dataset by giving similar weights to similar quests in the *latent factor* space. This can be realized by *individually* factorizing each slice of $\boldsymbol{X}_i$ into combination of two factor matrices (Furnas et al. 1988; Cremonesi, Koren, and Turrin 2010; Sifa, Bauckhage, and Drachen 2014) by considering the *truncated* Singular Value Decomposition as

$$\boldsymbol{X}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i^T = \boldsymbol{U}_i \boldsymbol{\Sigma}_i^{\frac{1}{2}} \boldsymbol{\Sigma}_i^{\frac{1}{2}} \boldsymbol{V}_i^T = \boldsymbol{Q}_i \boldsymbol{P}_i, \tag{2}$$

where for $k \leq rank(\boldsymbol{X}_i)$, the diagonal $\boldsymbol{\Sigma}_i \in \mathbb{R}^{k \times k}$ contains the value-sorted highest k singular values, $\boldsymbol{\Sigma}_i^{\frac{1}{2}}$ is its square rooted values, $\boldsymbol{U}_i \in \mathbb{R}^{m \times k}$ and $\boldsymbol{V}_i \in \mathbb{R}^{n \times k}$ are respectively the truncated left and right basis matrices, $\boldsymbol{Q}_i \in \mathbb{R}^{m \times k}$ is the factor matrix for the quests and $\boldsymbol{P}_i \in \mathbb{R}^{k \times n}$ is the factor matrix for the players. Similar to our neighborhood oriented recommendation approach, combining each individual player factor matrix as $\boldsymbol{\mathcal{P}} = \{\boldsymbol{P}_1, \ldots, \boldsymbol{P}_d\}$, we can build a recommender system by finding similarities between players by means of (1) for $\boldsymbol{\mathcal{P}}$ in the latent factor spaces (instead of $\boldsymbol{\mathcal{X}}$) corresponding to the players. In the following, we will refer to this recommendation method as the matrix factorization (MF) approach.

Introducing Tensor Factorization for Game Content Recommender Systems

Although our matrix factorization approach is designed to capture latent similarities between the available quests and players for each of our d behavioral aspects, it does not consider patterns among these aspects. This is especially important for our case as each slice of $\boldsymbol{\mathcal{X}}$ encodes different behavioral aspects of the *exact* same players. Tensor factorization methods tackle this issue by representing the data using local as well as global factors (Kolda and Bader 2009; Sifa et al. 2016). In this section we will introduce the tensor factorization model called Relaxed Tensor Dual Decomposition into Directed Components (RTDD), aka the TUCKER-II decomposition (Tucker 1966; Kolda and Bader 2009), derive an easy-to-implement algorithm to find its factors and finally show how we can use the resulting decomposition for game content recommendation.

Given a tensor containing bipartite column data matrices $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_d\}$, where $\boldsymbol{Y}_i \in \mathbb{R}^{m \times n} \; \forall \; 1 \leq i \leq d$, RTDD represents each slice of $\boldsymbol{\mathcal{Y}}$ as

$$\boldsymbol{Y}_i = \boldsymbol{A} \boldsymbol{W}_i \boldsymbol{B}^T \tag{3}$$

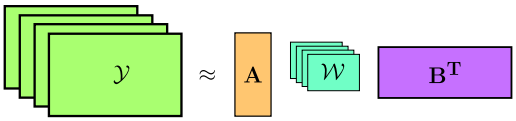


Figure 2: Generalization of DEDICOM for factorizing a collection bipartite matrices for game content recommendations. The model partitions each slice $\boldsymbol{Y}_i \in \mathbb{R}^{m\times n}$ of the data tensor $\boldsymbol{\mathcal{Y}} = \{\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_d\}$ as a combination of two global basis matrices $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ and $\boldsymbol{B} \in \mathbb{R}^{n\times q}$ and the corresponding local weighting matrix $\boldsymbol{W}_i \in \mathbb{R}^{p\times q}$.

where $\boldsymbol{A} \in \mathbb{R}^{m\times p}$ is the left basis matrix, $\boldsymbol{W}_i \in \mathbb{R}^{p\times q}$ is the coefficient matrix and $\boldsymbol{B} \in \mathbb{R}^{n\times q}$ is the right basis matrix. RTDD is a generalization of Dual Tensor DEDICOM, Tensor DEDICOM, and INDSCAL (Harshman 1978; Sifa et al. 2016) to provide low rank representations of collections of bipartite tensors, while the other methods work on square similarity matrices (see Fig.2).

Turning our attention to finding RTDD factors, we note that similar to Tensor DEDICOM and INDSCAL (Kolda and Bader 2009; Sifa et al. 2016), given a data tensor $\boldsymbol{\mathcal{Y}}$ and a set of desired factor parameters (p, q), RTDD factors can be found by minimizing the sum of the reconstruction error, which is usually represented as the sum of the squared matrix norm. Namely, we aim to find optimal factors $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{\mathcal{W}} = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_d\}$ minimizing

$$E_{RTDD}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\mathcal{W}}) = \sum_{i=1}^{d} \left\| \boldsymbol{Y}_i - \boldsymbol{A}\,\boldsymbol{W}_i\,\boldsymbol{B}^T \right\|^2. \quad (4)$$

It is important to note that, since (4) is not convex on the multiplied factors, optimal factors minimizing (4) are found in an alternating least squares (ALS) fashion (Kolda and Bader 2009; Sifa et al. 2016). In this case we iteratively minimize our error function over individual factors while keeping the others fixed.

We now turn our attention to deriving updates for an ALS algorithm minimizing (4), which we will group under three different types with respect to their algebraic structure for $\boldsymbol{A}$, $\boldsymbol{B}$ and each slice of $\boldsymbol{\mathcal{W}}$. We will particularly concentrate on updates constraining the basis matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ to be column orthonormal, i.e. $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}_p$ and $\boldsymbol{B}^T\boldsymbol{B} = \boldsymbol{I}_q$.

Similar to case for Tensor DEDICOM (Sifa et al. 2016), updates of the matrices $\boldsymbol{W}_i$ can be computed by considering $\boldsymbol{W}_i \leftarrow \boldsymbol{A}^{\dagger}\boldsymbol{Y}_i\boldsymbol{B}^{T\dagger} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}^T\boldsymbol{Y}_i\,\boldsymbol{B}\left(\boldsymbol{B}^T\boldsymbol{B}\right)^{-1}$ which, in case of orthogonal $\boldsymbol{A}$ and $\boldsymbol{B}$, simplifies to

$$\boldsymbol{W}_i \leftarrow \boldsymbol{A}^T\boldsymbol{Y}_i\,\boldsymbol{B}. \quad (5)$$

To derive an update for $\boldsymbol{A}$ and $\boldsymbol{B}$ we consider the (linearized (Sifa et al. 2016)) trace representation of (4)

$$E_{RTDD} = \sum_{i=1}^{d} \operatorname{tr}\left[\boldsymbol{Y}_i^T\boldsymbol{Y}_i\right] - 2\operatorname{tr}\left[\boldsymbol{Y}_i^T\boldsymbol{A}\boldsymbol{W}_i\boldsymbol{B}^T\right] + \operatorname{tr}\left[\boldsymbol{B}\boldsymbol{W}_i^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{W}_i\boldsymbol{B}^T\right] \quad (6)$$

and note that the gram matrix $\left[\boldsymbol{Y}_i^T\boldsymbol{Y}_i\right]$ neither depends on $\boldsymbol{A}$ nor $\boldsymbol{B}$. Additionally, the term $\operatorname{tr}\left[\boldsymbol{B}\boldsymbol{W}_i^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{W}_i\boldsymbol{B}^T\right]$ can be simplified because of orthogonality which leads to $\operatorname{tr}\left[\boldsymbol{B}\boldsymbol{W}_i^T\boldsymbol{W}_i\boldsymbol{B}^T\right]$. Using the invariance of traces under cyclic permutations leads to $\operatorname{tr}\left[\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{W}_i^T\boldsymbol{W}_i\right]$, which results in $\operatorname{tr}\left[\boldsymbol{W}_i^T\boldsymbol{W}_i\right]$ and is also independent of $\boldsymbol{A}$ and $\boldsymbol{B}$. Therefore, minimizing our reconstruction error in (6) with respect to $\boldsymbol{A}$ or $\boldsymbol{B}$ under orthogonality constraints is equivalent to *maximizing* the trace function

$$\hat{E}_{RTDD} = \sum_{i=1}^{d} \operatorname{tr}\left[\boldsymbol{Y}_i^T\boldsymbol{A}\boldsymbol{W}_i\boldsymbol{B}^T\right]. \quad (7)$$

So as to derive an ALS update for $\boldsymbol{A}$, we can isolate $\boldsymbol{A}$ for (7) by using the cyclic permutation and linearity of the traces as

$$\begin{aligned}\hat{E}_{RTDD} &= \sum_{i=1}^{d} \operatorname{tr}\left[\boldsymbol{A}\boldsymbol{W}_i\boldsymbol{B}^T\boldsymbol{Y}_i^T\right] \\ &= \operatorname{tr}\left[\boldsymbol{A}\sum_{i=1}^{d}\boldsymbol{W}_i\boldsymbol{B}^T\boldsymbol{Y}_i^T\right]. \end{aligned} \quad (8)$$

Considering $\boldsymbol{D} = \sum_i \boldsymbol{W}_i\boldsymbol{B}^T\boldsymbol{Y}_i^T$ and the *thin* SVD of its transpose (since $p \ll m$) $\boldsymbol{D}^T = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$, we can reformulate (8) as

$$\hat{E}_{RTDD} = \operatorname{tr}\left[\boldsymbol{A}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^T\right] = \operatorname{tr}\left[\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V}\boldsymbol{\Sigma}\right]. \quad (9)$$

We note the reformulation in (9) results in a special trace function of special type of multiplication of a semi-orthonormal matrix $\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V}$ and nonnegative diagonal matrix $\boldsymbol{\Sigma}$, whose upper bound is $\operatorname{tr}[\boldsymbol{\Sigma}]$ (Ten Berge 1983). Given that the maximum of (7) is attained for $\boldsymbol{U}^T\boldsymbol{A}\boldsymbol{V} = \boldsymbol{I}_p$ allows us to define an update for $\boldsymbol{A}$ minimizing (4) as

$$\boldsymbol{A} \leftarrow \boldsymbol{U}\,\boldsymbol{V}^T. \quad (10)$$

Finally, we can derive an ALS update for $\boldsymbol{B}$ to maximize (7) following the same steps as above. That is, considering the assignment $\boldsymbol{J} = \sum_i \boldsymbol{W}_i^T\boldsymbol{A}^T\boldsymbol{Y}_i$ and the SVD of its transpose as $\boldsymbol{J}^T = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^T$ results in an ALS update

$$\boldsymbol{B} \leftarrow \hat{\boldsymbol{U}}\,\hat{\boldsymbol{V}}^T. \quad (11)$$

In summary our ALS algorithm to factorize a given tensor $\boldsymbol{\mathcal{Y}}$ containing collection of bipartite matrices as in (3) by randomly initializing a tensor $\boldsymbol{\mathcal{W}}$ and orthogonal matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and respectively considering the updates we defined in (5), (10), (11), which defines one ALS iteration, until reaching a predefined number of iterations or a thresholded value for (4) (Sifa et al. 2016).

Considering our quest data tensor $\boldsymbol{\mathcal{X}}$ from the previous section, once we have found proper RTDD factors $\{\boldsymbol{A}, \boldsymbol{\mathcal{W}}, \boldsymbol{B}\}$ using our above algorithm to factorize $\boldsymbol{\mathcal{X}}$ as in (3), we can construct a player factor tensor $\hat{\boldsymbol{\mathcal{P}}} = \{\hat{\boldsymbol{P}}_1, \ldots, \hat{\boldsymbol{P}}_d\}$, where we define each slice of $\hat{\boldsymbol{\mathcal{P}}}$ as $\hat{\boldsymbol{P}}_c = \boldsymbol{W}_c\boldsymbol{B}^T$. Similar to our NO and MF approaches, our RTDD approach is based on finding similar players by means of our composite similarity measure from (1) using $\hat{\boldsymbol{\mathcal{P}}}$.

	Baseline	CB	NO	MF	RTDD
Day 1	8.21	8.64	7.79	8.92	8.93
Day 2	5.62	6.62	5.90	6.49	6.78
Day 3	3.96	4.64	4.40	4.55	4.74
Day 4	2.88	3.49	2.89	3.30	3.48
Day 5	2.02	2.44	1.82	1.94	2.30
Day 6	1.19	1.55	1.11	1.25	1.51
Day 7	0.25	0.54	0.61	0.47	0.72
Average	3.45	**3.99**	3.50	**3.85**	**4.07**

Table 1: Daily retention rates from our online bucket testing analysis for the first week of game play. The recommendation algorithms we compared in this work include random (Baseline), content-based (CB), neighborhood oriented (NO), and, matrix and tensor factorization based recommendation models (respectively MF and RTDD). Our results indicate that our tensor factorization based quest recommender has always resulted better than our baseline and had the best daily average values for retaining players in the game.

Recommendation Settings and Results

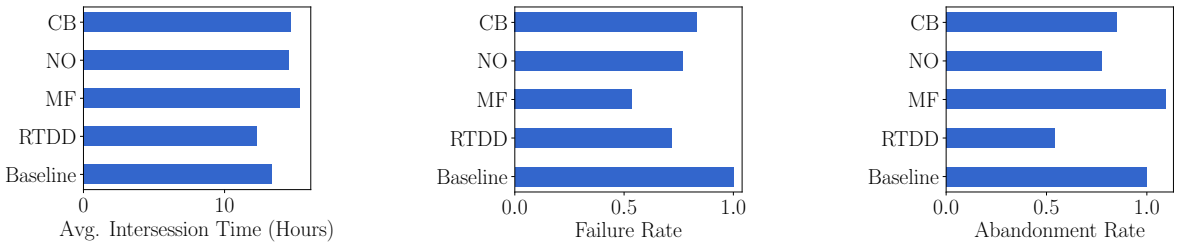
We first conducted a pilot study with a small number of players to test our platform during which, we fixed bugs and as mentioned, updated the failure rate heuristic of the CB. Following that, we evaluated our learning methods with a total of 243,279 players out of which 129,983 players were of interest since they played beyond the tutorial. Out of these players, we selected a cohort of randomly chosen 25,686 players for our bucket testing to benchmark our recommender algorithms. We have chosen the settings of our models using the offline data from the 104,297 players. The game was played on 12 different operating systems and three types of platforms that include phones, tablets and desktop computers. We have evaluated nine different groups (containing 2,854 players for each setting): considering our baseline recommender, that randomly assigns quests to players, as well as our content-based (CB), neighborhood oriented (NO), matrix and tensor factorization (respectively MF and RTDD) based recommendation algorithms. Each new user was assigned to a group via a round-robin approach and once assigned the players are never rotated to other groups.

Note that, since not every game aspect has the same significance, we considered individual weights when calculating the similarity in (1). For all of our data driven methods, we rate the user skill build highly so level, strength, agility and damage have a weightage of 25%, 10%, 10% and 10% respectively. While accuracy, health lost and completion time is 25%, 15% and 5% respectively. As the number of basis vectors for factor based models has a crucial impact on the recommendation quality (Cremonesi, Koren, and Turrin 2010; Sifa, Bauckhage, and Drachen 2014; Ryan et al. 2015a), we considered three different settings for both for MF and RTDD in our online evaluation. Note that, unlike our CB and NO methods our MF and RTDD models required us to compute and maintain the factorized matrices to support the filtering process. We found a factorization cycle of three hours to be a reasonable trade-off between capturing the different temporal behavior and handling the factorization process for an increasing number of players.

As for our evaluation metrics, we note that although being game dependent, success of free-to-play game titles usually depends on two different yet interlinked rates: *retention* and *monetization* (Runge et al. 2014; Sifa et al. 2015; Viljanen et al. 2018). Additionally, free-to-play games benefit from high retention rates to increase the likelihood of purchasing and clicking on in game ads and for that reason we will consider retention as our main evaluation metric of our recommendation algorithms. Given that, considering the daily average retention for the first week of gameplay, we note that in general our methods yielded better retention rates than our baseline, which was 3.45%. The retention rates for our CB and NO approaches were 3.99% and 3.50% respectively. For our MF recommender we have chosen the number of basis vectors $k \in \{100, 500, 2000\}$, which yielded the average retention rates of 3.58%, 3.85% and 3.38% respectively. Whereas, for RTDD we considered the numbers of basis vectors $(p, q) \in \{(700, 500), (1000, 2000), (2000, 4000)\}$ and obtained the retention rates of respectively 4.07%, 3.79%, 3.58%. We summarize our daily retention results of our best models in Tbl. 1. Another retention indicator in free-to-play games is related to the time between two consecutive sessions (Hadiji et al. 2014; Sifa et al. 2015) that developers aim to minimize. Analyzing the average intersession time values from our recommendation methods (see Fig. 3a), we note that only RTDD has yielded better results than our baseline method. Following that, recalling our goal of having an optimal difficulty level, we evaluated our methods considering the failure rate (the ratio of failed quests versus attempted quests) of the recommended quests, which we again aim to *ideally* minimize to reduce the likelihood of player frustration. Comparing the results from Fig. 3b, we observe that the recommendations of MF nearly halve the failure rates while the recommendations of RTDD reduce the failure rate by 28.43%. Together with our earlier results, this indicates that unlike the MF method, our RTDD based recommender system has recommended quests that are challenging enough to fail (more than the ones recommended from MF) yet more engaging due to better retention values. Similarly, our RTDD method reduced the abandonment rate (see Fig. 3c) of quests by 46% which implies that the recommended quests to the players fit their profile and are preferable for them.

Conclusion and Future Work

In this paper, we introduced a game content recommender system based on a tensor factorization approach. We compared our method to several baseline techniques using an online bucket testing evaluation based on extensive real world data to recommend in-game quests. Our results showed that tensor factorization based recommender systems overall performed better than complex, manually designed rule based systems, In contrast to the latter, our approach does not require experts to become involved but works in a purely data driven manner. Our future work will use these characteristics of our recommender system to adapt it to different forms of game content as well as to a wider variety of games.



(a) time between sessions

(b) failed quests

(c) abandoned quests

Figure 3: Summarizing our evaluation metrics of active users to evaluate the performance of the learning methods. (a) shows the average time in hours between user's play sessions, (b) compares the ratio of failed quests versus attempted quests and (c) shows the ratios of abandoned quests. For all the metrics the lower is better.

Acknowledgements

We would like to thank our anonymous reviewers for their insightful comments and suggestions. We would like to thank Flying Sheep Studios and the development team of *Trolljäger: Abenteuer in den Trollhöhlen* for creating the platform, providing us access to its analytics suite and supporting us with the evaluation process. Additionally, we would like to thank SRTL for supporting us to conduct this study. This research is supported by the Fraunhofer Center for Machine Learning.

References

Chen, J. 2007. Flow in games (and everything else). *Communications of the ACM* 50(4):31–34.

Cremonesi, P.; Koren, Y.; and Turrin, R. 2010. Performance of Recommender Algorithms on Top-N Recommendation Tasks. In *Proc. of ACM Recsys*.

Furnas, G. W.; Deerwester, S.; Dumais, S. T.; Landauer, T. K.; Harshman, R. A.; Streeter, L. A.; and Lochbaum, K. E. 1988. Information Retrieval Using a Singular Value Decomposition Model of Latent Semantic Structure. In *Proc. of ACM SIGIR*.

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

Harshman, R. A. 1978. 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*.

Kolda, T. G., and Bader, B. W. 2009. Tensor Decompositions and Applications. *SIAM Review* 51(3).

Leskovec, J.; Rajaraman, A.; and Ullman, J. D. 2014. *Recommendation Systems*. Cambridge University Press, 2 edition. 292–324.

Meidl, M.; Lytinen, S.; and Raison, K. 2014. Using Game Reviews to Recommend Games. In *Proc. AAAI AIIDE*.

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

Ryan, J. O.; Kaltman, E.; Hong, T.; Mateas, M.; and Wardrip-Fruin, N. 2015a. People Tend to Like Related Games. In *Proc. of FDG*.

Ryan, J. O.; Kaltman, E.; Mateas, M.; and Wardrip-Fruin, N. 2015b. What We Talk About When We Talk About Games: Bottom-up Game Studies Using Natural Language Processing. *In Proc. FDG*.

Saas, A.; Guitart, A.; and Perianez, A. 2016. Discovering Playing Patterns: Time Series Clustering of Free-To-Play Game Data. In *Proc. of IEEE CIG*.

Sifa, R.; Bauckhage, C.; and Drachen, A. 2014. Archetypal Game Recommender Systems. In *Proc. KDML-LWA*.

Sifa, R.; Hadiji, F.; Runge, J.; Drachen, A.; Kersting, K.; and Bauckhage, C. 2015. Predicting Purchase Decisions in Mobile Free-to-Play Games. In *Proc. of AAAI AIIDE*.

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

Sifa, R.; Pawlakos, E.; Zhai, K.; Haran, S.; Jha, R.; Klabjan, D.; and Drachen, A. 2018. Controlling the Crucible: A Novel PvP Recommender Systems Framework for Destiny. In *Proc. of ACM ACSW IE*.

Smith, B., and Linden, G. 2017. Two decades of recommender systems at Amazon.com. *Internet Computing* 21(3).

Ten Berge, J. M. 1983. A Generalization of Kristof's Theorem on the Trace of Certain Matrix Products. *Psychometrika* 48(4).

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

Viljanen, M.; Airola, A.; Heikkonen, J.; and Pahikkala, T. 2018. Playtime Measurement with Survival Analysis. *IEEE Transactions on Games*.

Weber, B. 2015. Building a Recommendation System for EverQuest Landmark's Marketplace. In GDC.

Zook, A.; Lee-Urban, S.; Drinkwater, M. R.; and Riedl, M. O. 2012. Skill-based mission generation: A data-driven temporal player modeling approach. In *Proc. of the Workshop on Procedural Content Generation in Games*. ACM.