A Sequential Recommendation Approach for Interactive Personalized Story Generation

Hong Yu and Mark O. Riedl
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{hong.yu, riedl}@cc.gatech.edu

ABSTRACT

In story-based games or other interactive story systems, a Drama Manager is an omniscient agent that acts to bring about a particular sequence of plot points for the user to experience. We present a Drama Manager that uses player modeling to personalize the user's story according to his or her storytelling preferences. In order to deliver personalized stories, a Drama Manager must make decisions on not only which plot points to be included into the unfolding story but also the optimal sequence of the events the user should experience. A prefix based collaborative filtering algorithm based on users' structural feedback is proposed to address the sequential selection problem. We demonstrate our system on a simple interactive story generation system based on choose-your-own-adventure stories to evaluate our algorithms. Results on human users and simulated users show that our Drama Manager is capable of capturing users' preference and generating personalized stories with high accuracy.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*games, Industrial automation*

General Terms

Algorithms, Design

Keywords

Interactive story generation, drama manager, player modeling, prefix based collaborative filtering

1. INTRODUCTION

Computer games use story to motivate player activity and to create a sense of causal continuity across a series of challenges [15]. While stories in games are often linear, progressively more games and virtual simulated environments allow variability in the story. A *Drama Manager* (DM) is an omniscient agent that monitors the virtual world in which the user is immersed and acts to determine what happens next in the player's story experience, often coordinating and/or instructing virtual characters [1]. In short, a Drama Manager is an agent that reasons about and attempts to enhance the user's experience in a virtual world.

Prevailing approaches to Drama Management develop agents that select successive plot points in response to player actions [9, 20, 14, 13, 7, 8]. In these systems, the DM is a surrogate for the human designer who provides some form of high-level specification for a "good" story experience. Although these action-response systems may improve player enjoyment, DM decisions do not take into account the user's preferences. We argue that the DM must also be a surrogate for the user by taking into consideration the user's preferences; the DM should model and act on the user's individualistic preferences do deliver optimized, personalized experiences.

Player modeling is a widely applied technique in computer games to capture users' preferences in order to increase enjoyment and reduce frustration and boredom [3]. Previous approaches to player modeling in story-based games have attempted to optimize player experience by classifying players according to well-defined player types [10, 4, 18]. and using pre-defined mappings of classes to plot point selection rules. These approaches require a designer to pre-determine the meaningful player types, even though there is no clear evidence of links between player types models and preferences for story content. User modeling can be cast as a recommendation process and collaborative filtering (CF) has been successfully applied to modeling user preferences over movies, products, books, music, etc [17]. Collaborative filtering algorithms attempt to detect users' rating "patterns," extract similarities between users' patterns, and make predictions of new user's ratings based on previous ratings from the users who share similar patterns. Collaborative filtering can be applied to the problem of selecting the plot point to occur next in a game, although to the best of our knowledge CF has not previously been used for Drama Management.

The techniques described above—player type classification and collaborative filtering— make one-shot recommendations and/or decisions. Unfortunately, stories are *sequences* of plot points and a player's assessment of the story he or she is experiencing is thus a function of the history of plot points experienced so far. In this paper, we present a novel approach to player modeling in games to select the most optimal sequence of plot points in a game: *Prefix Based Collaborative Filtering* (PBCF), which learns user preferences over fragments of story and then applies it to the Drama Management problem of selection of successive plot points in a game.

Appears in: *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Conitzer, Winikoff, Padgham, and van der Hoek (eds.), 4-8 June 2012, Valencia, Spain.

As a form of collaborative filtering, PBCF is a robust approach to player modeling in story-based computer games that uses machine learning to learn the most appropriate dimensions for the model from structured feedback—i.e., ratings of previously played story content. Player models can be built without making rigid assumptions about the model dimensions and those models will be more capable of describing, distinguishing, and capturing users' preferences. We show how a DM agent can make more effective decisions about how to act on the world for the benefit of the user, essentially optimizing the player's game experience. It is not enough to make a local decision about what should happen next in a game; a DM must determine the best possible future sequence of events the user should experience based on all previous events. Therefore, our DM agent, using PBCF, learns preferences over sequences of story events.

Our contributions are as follows: We (1) propose a new algorithm, PBCF, that performs sequential recommendation; (2) build a flexible and robust preference model that does not rely on assumptions about the model dimensions; (3) incorporate the preference model into the Drama Manager which is allowed to select plot points based on users' preferences instead of user actions.We evaluate the algorithm in a simplified testbed domain based on Choose-Your-Own-Adventure book serials[1] using human users and simulated users.

2. RELATED WORK

Drama Manager agents have been widely used to guide the users through an expected story experience set by designers. Two approaches to drama management—search-based drama management [20, 9, 16] and declarative optimization-based drama management [14, 2]—transform the problem of selecting the next best plot point into a search problem where the DM searches for possible future histories of plot points based on an evaluation function set by the designer. The Façade interactive drama [8] uses a reactive plot point selection technique to determine the next set of behaviors for two virtual characters. Riedl and colleagues [13] use a partial-order planner to re-plan a story when the user performs actions that change the virtual world in ways that prevent story progression as expected. Similarly, Porteous and Cavazza [11] use a planner with designer-provided constraints to control virtual characters and push a story forward. A DM by Magerko et al. [7] predicts player actions and attempts to prevent story failures by directing virtual characters to perform actions or change goals. These Drama Management techniques all respond to player actions to move the story forward in a way partially or completely conceived by a human designer. That is, the DM is a surrogate for the human designer.

Relatively little work has been done to determine how a story should unfold in a game or virtual environment based on player models. The PaSSAGE system [18] automatically learns a model of the player's preference through observations of the player in the virtual world, and uses the model to dynamically select the branches of a Choose-Your-Own-Adventure style story graph. PaSSAGE uses Robin's Laws five game player types: Fighters, Power Gamers, Tacticians, Storyteller, and Method Actors. A player is modeled as a vector where each dimension is the strength of one of the types. As the player performs actions, dimensions are increased or decreased in accordance to rules. Peinado and Gervás [10] use the same player types. Seif El-Nasr [5] uses a four-dimension player model: heroism, violence, self-interestedness, and cowardice. These player modeling techniques assume players can be classified according to several discrete play styles and that, even with continuous characteristic vector combining the discrete user styles, optimal story choices can be made by a DM. These systems further assume that role playing game player classifications (or ad-hoc types) are applicable to story plot choices. In addition, these systems assume that plot points could be selected in isolation from each other based on a comparison between their attributes and the player model. In this paper, we propose a collaborative filtering based player modeling approach that learns player model dimensions from user feedback—ratings—and further solves sequential plot point recommendation/selection problems.

Roberts, et al. [14, 2] developed an algorithm, Targeted Trajectory Distribution Markov Decision Process (TTD-MDP), to solve non-Markov Decision Processes by wrapping all the previous MDP states into one node of a trajectory tree. Their objective was to produce probabilistic policies for the trajectory tree that minimize divergence from a target distribution of trajectories. They apply their process to declarative optimization-based drama management by modeling stories as state space trajectories. TTD-MDPs require a target distribution across trajectories/stories. Further, as a reinforcement learning technique, it must simulate a user. While the simulated user may utilize a player model, that model would need to first be acquired. Our approach learns the player model and does not require a target distribution over trajectories.

3. PREFIX BASED COLLABORATIVE FILTERING

A story can be decomposed into a sequence of plot points, which usually represent single events or tasks in the story. In a interactive story-based game or virtual world, a Drama Manager is in charge of which plot points to be presented to the users and in what order. This is a NP complete problem given all the plot points. To address this, temporal and semantic constraints are imposed among these plot points to reduce the size of the story space [20, 9]. When constraints are known, a *branching story graph* containing the possible successors to each plot point can be built automatically or by hand. The question a DM must answer is: what is the best path in the branching story graph for an individual user? By answering the quest with a player model, we aim to create an optimal experience for the a particular user.

The architecture of our Drama Manager system is illustrated in Figure 1. The DM has an existing story library or database which contains all possible story permutations, as described in the next section. The DM obtains the current system states from the interactive system interface. Then the best path is selected by the DM according to the player model, which is built entirely based on player feedback.

3.1 Story Representation

A branching story graph is a representation that specifies which plot points are allowed to follow other plot points. For the purposes of a Drama Manager agent, it provides

[1]http://en.wikipedia.org/wiki/Choose_Your_Own_Adventure

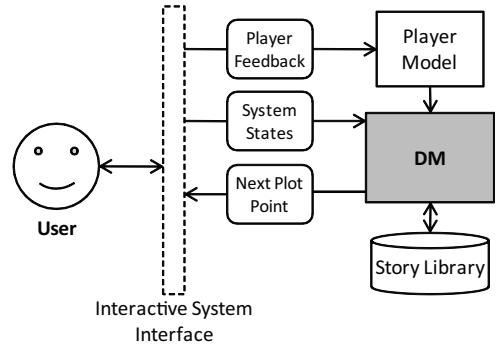

Figure 1: The architecture of the interactive story generation system.

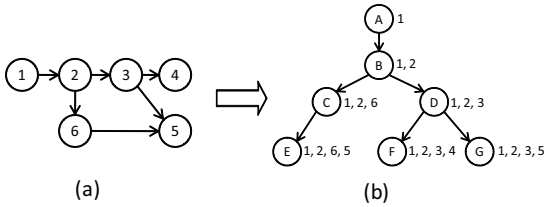

Figure 2: (a) Branching story graph of a simple story library which contains three stories. (b) The prefix graph of the story library.

the set of options for the next plot point at any given time. Figure 2(a) illustrates a very simple branching story graph, where nodes represent plot points and links represent possible alternative successors. While simple, many other plot representations are reducible to the branching story graph representation [12, 14]. A story is a path through the graph starting at the root node and terminating at a leaf node. Figure 2(a) contains three complete, possible stories ({1,2,3,4}, {1,2,3,5}, {1,2,6,5}). Note that the branching story graph is usually a graph instead of a tree.

We transform it into a prefix graph as in Figure 2(b). In the prefix graph, each node represents a prefix of a story. The children of a node are those prefixes that can directly follow the parent prefix. Apparently, the prefix graph will be a tree or forest since any prefix cannot have more than one parent node. In our system, only the prefix graph is stored in the story library.

In our approach, we ask users to rate the "story-so-far"—the portion of the story that they have observed leading up to the current point in time. Notice that it is easier and more accurate for the users to rate the story-so-far instead of each new plot point since history matters in stories. Because the branching story graph has been transformed into a prefix graph, the rating is stored with the prefix that corresponds to the story-so-far. Further, the system does not need to solve a credit assignment problem as in reinforcement learning to determine how much of a final rating each plot point is responsible for.

3.2 Player Modeling

Different users can different preference over stories. We aim to extract the dimensions of these preferences from the users' ratings instead of constraining ourselves to a few pre-defined dimensions as in prior works by Thue et al., [18], Peinado et al. [10], and [4]. The basic assumption of our player modeling algorithms is that those people who share similar preference in the past tend to share it again in the future, hence we use a form of collaborative filtering.

Prefix	User 1	User 2	User 3	...
A (1)	*	*	2	...
B (1, 2)	1	*	2	....
C (1, 2, 6)	*	*	*	...
D (1, 2, 3)	4	3	*	...
...	...	...	...	...

Figure 3: Illustration of the prefix-rating matrix. *A*, *B*, *C* and *D* represent the prefixes. The larger the digital number, the higher the preference. The stars represent those missing ratings.

Unlike in traditional recommendation systems, where collaborative filtering is usually used as one-shot recommendation of content based on preferences, we must solve the problem of sequences of recommendations where the order of plot points matters. In other words, the story generation can be viewed as as a non-Markov Decision Process, where at each step the DM's selection of optimal next plot point is based on all previous plot points. For example, if the story prefix {1, 2, 3} has been presented to the user (node D in Figure 2(b)), then the DM's next selection should be based on all the user's ratings on previous three prefix nodes (A, B, and D). A user who leaves positive feedback on node B and negative feedback on node D should be different from another one who leaves negative feedback on both node B and D. A prefix based CF approach is proposed to model players in a way that allows non-MDP problems to be solved. While other techniques similarly roll history into state nodes, as in our prefix trees and equivalent structures in TTD-MDPs [14, 2], our approach uses structured feedback to guide tree navigation, inferring ratings when feedback data is sparse.

In this paper, stories are presented to the user plot point by plot point and a preference rating for the story-so-far is collected after every plot point. Then a *prefix-rating matrix* including the story prefix ratings from all users can be obtained. A n by m prefix-rating matrix contains the ratings for n prefixes from m users. One column of the matrix represents all the ratings of the corresponding user for the all the prefixes. One row of the matrix represents ratings for the corresponding prefix from all the users. Figure 3 shows a simple illustration of the prefix-user matrix. The matrix is usually very sparse, i.e. containing a lot of missing ratings, because there is no way of expecting any given user to have read and rated all the prefixes in the library. The prefix-user matrix is treated as the product-user matrix as in traditional CF [17].

Two collaborative filtering learning algorithms are tested in this paper: *probabilistic Principle Component Analysis* (pPCA) [19] and *Non-negative Matrix Factorization* (NMF) [6, 21]. Next we briefly introduce the two algorithms and their application in our player modeling.

3.2.1 Probabilistic PCA

For a n dimensional vector $\mathbf{r}$, probabilistic PCA assumes

that it can be factorized as follows:

$$\mathbf{r} = W\mathbf{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \quad (1)$$

where $\mathbf{x}$ is a n' dimensional vector in the hidden or reduced dimension space (usually $n' < n$) and W is a n by n' matrix. $\boldsymbol{\mu}$ is the mean vector which permits $\mathbf{r}$ to have nonzero mean. $\boldsymbol{\epsilon} \sim \sigma^2\mathrm{I}$ is the Gaussian noise.

Let the vector $\mathbf{r}$ be any one column of the prefix-rating matrix. pPCA projects the corresponding user's prefix-rating vector into the hidden space or the reduced dimension space $\mathbf{x}$ just as in traditional principle component analysis. The hidden space vector $\mathbf{x}$ models the corresponding user's preference type. Note that from Equation 1 we can get:

$$\mathbf{r}|\mathbf{x} \sim N(W\mathbf{x} + \boldsymbol{\mu}, \sigma^2\mathrm{I}) \quad (2)$$

Thus the basic assumption of pPCA algorithm is that the user's prefix rating vector (the column of the prefix-user matrix) obeys a multi-dimensional Gaussian distribution.

If the prefix-user matrix contains missing values, the EM algorithm can be used to compute W and σ [19]. The hidden space vector $\mathbf{x}$ can then be computed from the observed ratings, and the missing ratings can be estimated with W, σ and $\mathbf{x}$.

3.2.2 Non-negative Matrix Factorization

The purpose of NMF is to factorize an n by m matrix R as follows:

$$R = W * H \quad (3)$$

where $W \in \mathbb{R}^{n*m'}$ and $H \in \mathbb{R}^{m'*m}$ are two non-negative matrices (usually $m' < m$). Non-negative here means that all the entries in the matrix are greater than or equal to zero. If R contains missing values, the EM algorithms can be used to compute W and H [21]. Then the missing values in R are recovered using the estimated W and H.

If R is the prefix-rating matrix (n prefixes and m users), the m' columns of the matrix W, $\boldsymbol{w}^j$ $j = 1, ...m'$, can be viewed as a set of bases that represent different types of users. Then $\boldsymbol{h}^i$, the i^{th} column of H, will correspond to the i^{th} user's preference. In practice, it will be difficult to interpret the player types that correspond to each $\boldsymbol{h}^i$. However, if we have prior knowledge about some preference types (e.g., fighter, tactician), that is, we know their ratings for all the prefixes (e.g., fighter's rating vector $\boldsymbol{w}^f$, tactician's rating vector $\boldsymbol{w}^t$), then the matrix W can be seeded with the rating vectors ($\boldsymbol{w}^f$, $\boldsymbol{w}^t$) as fixed columns. Simulated experiments in Section 4 shows that such prior can indeed increase player modeling accuracy when they are known to accurately distinguish users with regard to stories.

3.3 Player Modeling Processes

The entire Drama Management system is composed of two phases: model training and story generation. In the model training phase, the process can be summarized as follows:

1. Build the story library storing the prefix forest.
2. Collect data and populate the prefix-rating matrix R.
3. Compute the player model parameters: W, σ and $\boldsymbol{\mu}$ for the pPCA, or W for NMF.

For a new user, after we get some initial ratings $\boldsymbol{r}$ from him or her, the story generation phase is as follows:

1. Model the new user's preference using $\boldsymbol{r}$ through computing $\boldsymbol{x}$ for pPCA, or $\boldsymbol{h}$ for NMF.
2. Calculate the full rating vector $\boldsymbol{r}'$ (with no missing values) from $\boldsymbol{x}$ using Equation 1 or Equation 3.
3. Select the highest rated full-length story that is a descendant of the current prefix in the prefix graph. Present the corresponding next plot point in the selected full-length story.
4. Collect user's rating on the story-so-far (i.e., the recommended prefix).
5. Include the new rating into $\boldsymbol{r}$ and go to step 1.

Note that it is not necessary to collect ratings after each prefix in the story generation phase; we do it in our system for the purpose of collecting as much data as possible to build a more accurate player model. With every new rating, the DM will get better understanding of the current user's preference and recommend next prefixes with higher confidence.

4. EVALUATION AND RESULTS

Based on the theory that all interactive systems can be translated into a branching story graph, we built a simple interactive storytelling system that can automatically guide the user through a particular branching path. The system presents the stories to the user one plot point by one plot point. After every plot point, it asks the user for the preference rating on the story-so-far. Instead of the user choosing the branch, the system then recommends the next branch by some means (using our model, or random). In the system, the ratings are digital integers ranging from 1 to 5. While our test bed is simple compared to modern computer games, it represents the fundamentals of other Drama Managers. By limiting player interaction to providing ratings of the story-so-far we aim to control the experimental variable of player agency to further facilitate validation of our preference model. We have performed two sets of experiments: on human users and on simulated users.

4.1 Story Library

The story library is built through transcribing the stories from four Choose-Your-Own-Adventure books—*The Abominable Snowman*, *Journey Under the Sea*, *Space and Beyond*, and *The Lost Jewels of Nabooti*—all of which are adventure stories. Every book contains a branching story tree. At the end of each page in the book, the reader is presented with a multi-choice question, the answer to which leads the reader to different pages of the book to continue down different branches of the story. Figure 4 shows the branching story graph from one of the books.

We chose to transcribe Choose-Your-Own-Adventure books to control for story quality, as opposed to authoring stories ourselves. In the system, every story is pruned and transcribed to contain exactly six plot points[2]. As in Figure 2, these branching story graphs are transformed into the prefix graphs which are stored in the story library. Thus our story library is a forest containing 154 possible stories (about 1000 words per story) and 326 prefixes.

[2]We do this for implementation purpose. It is not necessary for every story to contain exactly the same number of plot points; our system can be easily extended to handle stories of varied number of plot points.

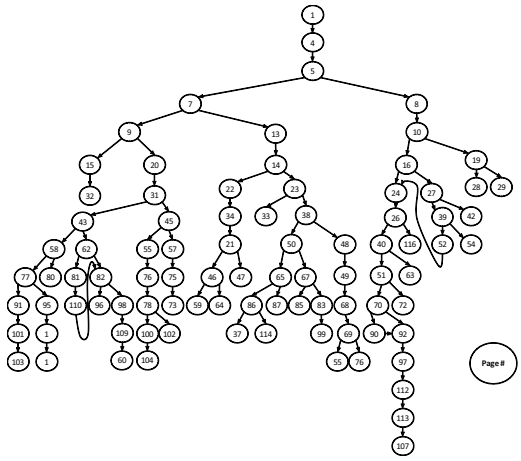

Figure 4: Illustration of the branching story representation of stories in *The Abominable Snowman*. The nodes in the graph represent pages in the book (plot points). Every story start from the root node and end on one of the leaves.

4.2 Experiments with Human Users

We conducted an evaluation of our system on human users. The evaluation consisted of two phases: model training, and story generation testing. In the model training phase, 31 users have participated in the experiments (18 male and 13 female). 26 of them are college graduate students and the other 5 users are research scientists and staff. All the users who have never read the choose-your-own-adventure stories are given a sample adventure story which is out of the story library to familiarize themselves (five out of the 31 users have been exposed to choose-your-own-adventure series before the experiments).

Every user in the training phase read ten stories randomly selected from the library. A random story is a random walk from a root of any tree in the forest to a leaf node. Each story is presented to the user one plot point at a time and a rating is collected after every plot point. The experiment took about half an hour for each player. We obtain a 326 by 31 prefix-rating matrix R with $\sim 86\%$ ratings missing.

We computed the Root Mean Square Error (RMSE) in order to get the best parameters for model training. The prefix-rating matrix R is randomly split into training part R^t which contains 90% of the ratings, and validation part R^v which contains the remaining 10% of the ratings. Note that R^t and R^v are still the same dimension as the original matrix R and both of them contains missing values. We train the NMF and pPCA on the training set R^t with different parameters. The resulting models are used to predict the ratings in the validation matrix. The Root Mean Square Error (RMSE) can be computed as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|O|}\sum_{i,j\in O}(R^v_{ij} - R^{v'}_{ij})^2} \tag{4}$$

where $R^{v'}$ is the predicted validation matrix, O is the set of entries indices that are not missing in the validation matrix R^v and $|O|$ is the number of entries that are not missing in R^v. The random splitting process is repeated for ten times and the average RMSEs on the validation sets are reported in Table 1. The dim$i$ in the table mean the number of columns of the matrix W in Equation 3 is i. The RMSEs in the table suggest that there are probably six types of users in current training set when it comes to story preferences.

Algorithms	RMSE
NMF_dim3	1.2423
NMF_dim4	1.1781
NMF_dim5	1.1371
NMF_dim6	0.9901
NMF_dim7	1.1108
NMF_dim8	1.1354
NMF_dim9	1.2464
pPCA	1.2016

Table 1: The average RMSE for different parameters of NMF and pPCA algorithms. NMF_dim$i$ means NMF algorithm with the number of player styles (number of columns of the matrix W) i.

	Random	Personalized	Accuracy	p-value
All	2.9449	3.8899	0.828	< 0.0001
Existing	3.032	4.035	0.863	< 0.0224
New	2.8993	3.8138	0.809	< 0.0001

Table 2: The average ratings for the random and personalized full-length stories. The accuracies are the percent of pairs in which the average rating of the personalized stories is larger than the average rating of the random stories.

Another 22 graduate students (17 male and 5 female) were recruited for the second phase: testing of the model's ability to predict ratings. This phase is divided into four steps. In the first step, the users read five random stories and leave ratings after every plot point, as in the training phase. In the second step, the DM starts to choose new personalized stories and branches according to the users' ratings and the player models built in the training phase. The final NMF and pPCA models are trained on the entire prefix-rating matrix R with the best parameters which correspond to the least RMSE. These personalized stories are presented to the users plot point by plot point in the same way as the first five random stories and the users' ratings after every plot point are collected. Then, as in Sharma et al. [16], the DM then presents another five personalized stories in the third step, followed by five random stories in the last step in order to eliminate any prejudice introduced by the order in which the stories are presented to the users. Thus, every user in the testing phase is required to read 20 stories (10 total random stories and 10 total personalized stories).

For comparison of model performance on new users versus existing users, we also invited 11 participants from the training phase back to also participate in the validation phase. The experiment process is exactly the same as above. Table 2 shows the results for the new users and existing users when the player model is trained with NMF algorithms set for six dimensions (the variant with the lowest RMSE).

Results are shown in Table 2, the first line exhibits the statistical results on all the 32 testing users. The second line and the third line give the results of the 11 users from the

training group and the 21 users out of the training group respectively. The first column "Random" and the second column "Personalized" show the average ratings of all the random and all the personalized stories in the story generation phase respectively. For every user in the story generation phase, we also compare the pair of average story ratings from the first step and the second step, and the pair of average story ratings from the third and the fourth step. The "Accuracy" column shows the percent of pairs in which the average rating of the personalized stories is larger than the average rating of the random stories, indicating the DM correctly choosing preferred stories. The last column shows the significance of the difference between random and personalized averages using a one-tailed t-test.

4.3 Experiments with Simulated Users

In order to establish more complete analysis on PBCF, we also conducted experiments with simulated users. Simulated users are more consistent over time, allowing us to make observations about our algorithm on a controllable data set to study its capability. Note that it is not necessary for the simulated users actually being good imitations of the human users' preferences. Instead, as long as the simulated users are consistent, they can be used to experiment with the capability of our system to capture users' preference and build user models. In addition, we can get as much rating data as we need from simulated users.

The simulated users are built based on the Robin's Laws player types, which assumes there are five types of players for games: Fighters (who prefer combat), Power Gamers (who prefer gaining items and riches), Tacticians (who prefer thinking creatively), Storytellers (who prefer complex plots) and Method Actors (who prefer to take dramatic actions) [18, 10]. Every simulated user is assigned with a five-dimensional characteristic vector. Each entry of the vector (ranging from 0 to 1) specifies the corresponding characteristic of the simulated user. For example, vector $[0.8, 0, 0, 0.6, 0]$ means the simulated user is a combination of fighter and storyteller and tends to enjoy fighting a little more.

To run experiments with simulated users, all story prefixes in our database were labeled according to Robin's Laws player types. The labeling of prefixes was performed by three college students, one of whom is an author on this paper. To mitigate bias, the label for each prefix is the average label produced by each of the human labelers. Thus each prefix label is a five-dimensional vector, where each element expresses the average belief about how the story-so-far matches players of different types.

We assume that a simulated user of a particular type according to the Robin's Laws player types tends to prefer a story or story prefix that most closely matches the user's type. For example, a simulated user with characteristic vector $\mathbf{u} = [0.8, 0, 0, 0, 0]$ will prefer for a story prefix i with label $\mathbf{p}_i = [1, 0, 0, 0, 0]$ over a story prefix j with label $\mathbf{p}_j = [0, 1, 0, 0, 0]$. Consequently, we assume that the rating r of a simulated user $\mathbf{u}$ for a prefix $\mathbf{p}$ is proportional to cosine distance between the vector $\mathbf{u}$ and $\mathbf{p}$: $r \sim \frac{\mathbf{u}^T\mathbf{p}}{|\mathbf{u}||\mathbf{p}|}$. In practice, the ratings are computed by scaling the cosine distances to between 1 and 5. In addition, we add random noise with standard Gaussian distribution (mean 0 and variance 1) to all the ratings in order to simulate the human user case where it could be inaccurate for the human users to quantitate preference into digital labels.

During the model training phase of the experiments, we generate 120 simulated users with characteristic vectors randomly chosen from {[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]}. Each simulated user then reads 10 random stories and leaves a preference rating after every plot points as human users. A 326 by 120 prefix-rating matrix is populated at the end of the model training phase.

The testing phase is divided into four steps that are exactly the same as the human user's experiment story generation phase. In every step each simulated user is required to read five random or personalized stories. To test the generalization ability of our algorithm, a new group of 1000 simulated users are used in the testing phase. Each is assigned with a random characteristic vector (the five entries of the characteristic vector are values ranging from 0 to 1).

For the purpose of comparison, we implement all the following algorithms:

- *BaselineP*: pPCA is used to learn the player models from simulated users' ratings on full-length stories instead of prefixes, then directly recommends the full-length stories instead of choosing branches through recommending prefixes. This algorithm behaves similar to a traditional movie recommendation system where full-length movies are recommended based on others' ratings on the full-length movies.
- *BaselineN*: The same as *BaselineP* except using NMF.
- *Vector*: A vector based player modeling algorithm that is similar to the model learning technique used by Thue et al. [18]. A vector that simulates a player starts out as $[0, 0, 0, 0, 0]$. For every plot point encountered, the DM updates the characteristic vector based on the features of the current story prefix including the new plot point. The DM generates successive plot points by recommending the following prefix based on the updated user vector, or chooses randomly when there is no clear preference.
- *pPCA*: The prefix based algorithm using pPCA, same as with the human users.
- *NMFwoP*: The prefix based algorithm using NMF without prior knowledge, same as with the human users.
- *NMFwP*: The prefix based algorithm using NMF with Robin's Laws player types as prior knowledge as discussed in Section 3.2.2. In the case of simulated users, we can compute the correct rating vector $\boldsymbol{w}^j$ for each known player type j, where $j = 1, ...5$ correspond to the five player types in the training phase. Then these vectors $\boldsymbol{w}^j$ can be included into the matrix W in Equation 3 as fixed columns during training process.

The experiment results for these algorithms on the 1000 testing simulated users are shown in Table 3. The results are all statistically significant at p-values approaching zero (using one-tailed t-tests on random and personalized averages) due to the large number of testing users.

It is interesting to explore the learning speed of the player model as the number of stories read in every step changes, which was set to 5 for testing with human and synthetic users. Figure 5 shows the average accuracies of 1000 simulated users for different algorithms as the number of stories read in every step changes. As shown in the figure, the NMF algorithms can achieve accuracies higher than 70% even when one new simulated user reads only one story.

Algorithm	Random	Personalized	Accuracy
BaselineP	2.2190	2.5305	0.668
BaselineN	2.1752	2.4582	0.643
Vector	2.2010	2.8335	0.617
pPCA	2.2350	2.9607	0.798
NMFwoP	2.2362	3.3950	0.894
NMFwP	2.2013	4.0027	0.949

Table 3: The testing results for the simulated users using several variations of the DM algorithm.

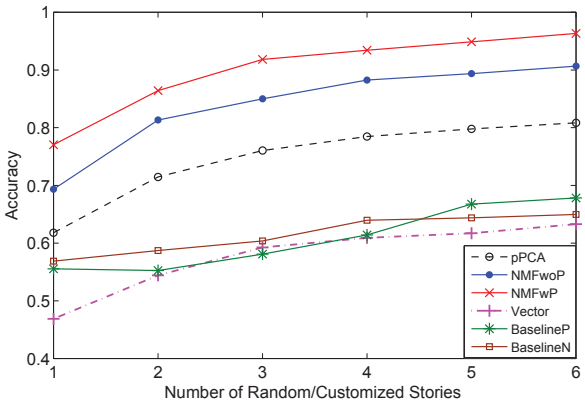

Figure 5: The accuracies of the six algorithms as the number of stories read in every step changes.

We also test the influence of the training set size on the player model training process. Figure 6 shows the average RMSEs of the three prefix based algorithms with different number of simulated users for training. Each RMSE value in the figure is an average computed from 10 random splittings of the training data. As seen from the figure, the training RMSEs decrease as the the training set size grows. Due to the Gaussian noise in the rating data, the RMSE values for the NMFwP algorithm become stable after the number of training users goes above 100 even if it has the perfect prior knowledge of the player models.

4.4 Discussions

Both the experiments on human users and simulated users achieve high story generation accuracies on the current Choose-Your-Own-Adventure data set for the prefix based algorithms. We observe that over 80% of the time, new human users will rate DM-generated stories higher than random stories. We achieve this rate after the new users have only rated 5 sample stories. The accuracy of about 86% is achieved when the testing users' data are already part of the trained model. The average ratings for the personalized stories are higher than the random stories. The results show that our prefix based player modeling algorithms can capture the users' preference and generate new stories with high confidence.

Even with Gaussian noise added to the synthetic ratings, our player modeling algorithms achieve higher accuracies on the simulated users than on the human users. Although from the learning process we know that there does exist features of story rating behavior that are predictive of future rating behavior, it is still difficult if not impossible to interpret

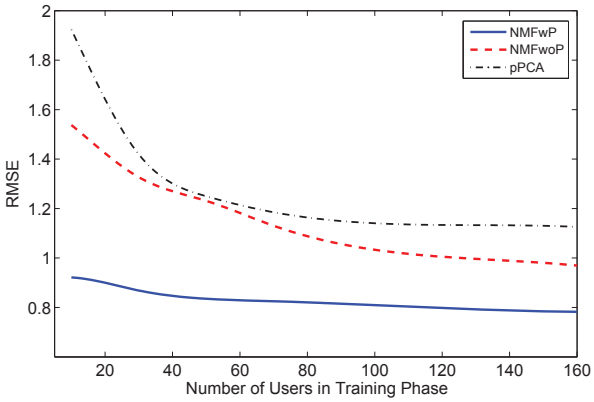

Figure 6: The average RMSEs of the three prefix based algorithms with different number of simulated users for training.

these features. We do not believe that these features are as clear cut as Robin's Laws player types, Seif El-Nasr's types, or other factorized personality models. The preference types of human users should be more complicated than the linear combinations of several presumed categories, which is also the reason to build the player models from data instead of constraining ourselves to a few pre-defined dimensions.

For the simulated users, the NMF algorithm usually performs better than the pPCA algorithm which could due to our linear model assumption for the simulated users. The linear characteristic model for simulated users coincides with the basic assumption of the NMF algorithm, which also assumes that users (columns of the matrix R in Equation 3) are linear combinations of a set of bases (columns of the matrix W in Equation 3). Although NMF is a natural fit for the synthetic users, it is also superior to pPCA for human data in terms of RMSEs in our experiments.

Figure 5 shows that the prefix based algorithms can acquire player preference much faster than applying traditional CF algorithms directly on full-length stories (baseline algorithms *BaselineP* and *BaselineN*). The main reason is that the prefix based algorithms can obtain more preference information (the ratings on all the prefixes) from users than the baseline algorithms in both model learning and story generation phases. It also demonstrates that these ratings on prefixes do strongly correlate with users' preference and can help to improve player models. The figure also shows that the Vector approach learns the player model much slower than our algorithms and is thus less accurate on average. This is because the Vector approach cannot acquire any information from the training data.

Although there are only 154 full-length stories and 326 prefixes in the story library, the well-known scalability of collaborative filtering algorithms suggests that our algorithms can be extended to handle larger scale problems as long as we have enough rating data. In traditional recommendation systems, CF algorithms can easily process products-user matrix with dimension of hundreds of thousands and achieve high recommendation accuracies [17]. The number of prefixes could be exponential in the number of plot points. But in practice we can effectively add constraints between plot points to limit the size of the prefix database. Notice that in our system, the number of total prefixes grows linearly with

the number of total stories given a limit of maximum number of plot points in each story because of the constraints imposed by the branching story graph representation.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a PBCF algorithm to address the sequential recommendation problem. The player models built with the PBCF algorithm enable the Drama Manager to make successive story choices on behalf of the users. We examine the algorithm in a simple interactive story generation system built with choose-your-own-adventure series. The evaluation results on both human users and simulated users demonstrate that our algorithm is able to capture the users' preference and generate stories with high confidence.

Although the current system is simple, it represents one of the most important fundamentals of Drama Management: delivering new stories based on a pre-authored library of legal stories through player modeling. Although player modeling techniques in Drama Management has not been widely explored, we believe that it is an essential part of building an agent that is responsible for optimizing the player's experience in a game or virtual world. Our approach combines the robustness of machine learning with intuitions about the sequential nature of stories. The Drama Manager agent built in this way is capable of generating personalized stories and guiding the users through a better experience.

There are many avenues for future work. A larger story space is required to verify the scalability of our algorithms. The experiment results on simulated user show that prior knowledge does help to increase the accuracies. It will be interesting to investigate how to include prior knowledge in the case of human users. Furthermore, the player choices and other unstructured feedback are still missing in the current system. The DM operates like a story generator in current system, although we implement this as a control to validate the algorithm. When the algorithm is placed in a full virtual world that supports human player choice, then other well-known techniques for managing player choice, such as re-planning [13] can be added to the player modeling.

6. REFERENCES

[1] J. Bates. Virtual reality, art, and entertainment. *Presence: The Journal of Tele-operators and Virtual Environments*, 1(1):133–138, 1992.

[2] S. Bhat, D. L. Roberts, M. J. Nelson, C. L. Isbell, and M. Mateas. A globally optimal algorithm for TTD-MDPs. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.

[3] D. Charles, M. McNeill, M. McAlister, M. Black, A. Moore, K. Stringer, J. Kĺźcklich, and A. Kerr. Player-centred game design: Player modelling and adaptive digital games. *Proceedings of DiGRA 2005 Conference*, 2005.

[4] M. S. El-Nasr. Interactive narrative architecture based on filmmaking theory. *International Journal on Intelligent Games and Simulation*, 3(1), 2004.

[5] M. S. El-Nasr. Engagement, interaction, and drama creating an engaging interactive narrative using performance arts theories. *Interactions Studies*, 8(2):209–240, 2007.

[6] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[7] B. Magerko and J. E. Laird. Mediating the tension between plot and interaction. *AAAI Workshop Series: Challenges in Game Artificial Intelligence*, 2004.

[8] M. Mateas and A. Stern. Integrating plot, character and natural language processing in the interactive drama facade. *Proceedings of the 1st International Conference on technologies for Interactive Digital Storytelling and Entertainment*, 2003.

[9] M. J. Nelson and M. Mateas. Search-based drama management in the interactive fiction Anchorhead. *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, 2005.

[10] F. Peinado and P. Gervas. Transferring game mastering laws to interactive digital storytelling. *Proceedings of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, 2004.

[11] J. Porteous and M. Cavazza. Controlling narrative generation with planning trajectories: the role of constraints. *Proceedings of 2nd International Conference on Interactive Digital Storytelling*, 2009.

[12] M. Riedl and R. M. Young. From linear story generation to branching story graphs. *IEEE Journal of Computer Graphics and Animation*, 26(3):23–31, 2006.

[13] M. O. Riedl, A. Stern, D. M. Dini, and J. M. Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications*, 2008.

[14] D. L. Roberts, M. J. Nelson, C. L. Isbell, M. Mateas, and M. L. Littman. Targeting specific distributions of trajectories in MDPs. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 2006.

[15] A. Rollings and E. Adams. Andrew rollings and ernest adams on game design. *New Riders Press*, 2003.

[16] M. Sharma, M. Mehta, S. Ontanon, and A. Ram. Player modeling evaluation for interactive fiction. *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment conference*, 2007.

[17] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[18] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen. Interactive storytelling: A player modelling approach. *Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment Conference*, 2007.

[19] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, B61(3):611–622, 1999.

[20] P. W. Weyhrauch. Guiding interactive drama. *Ph.D. Dissertation*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-97-109, 1997.

[21] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. *Proceedings of the 6th SIAM Conference on Data Mining*, 2006.