Pre-print of an article to appear in the IEEE Transactions on Computational Intelligence and Artificial Intelligence in Games (in press).

Personalized Interactive Narratives via Sequential Recommendation of Plot Points

Hong Yu and Mark O. Riedl

***Abstract*—In story-based games or other interactive systems, a drama manager is an omniscient agent that acts to bring about a particular sequence of plot points for the player to experience. Traditionally, the drama manager's narrative evaluation criteria are solely derived from a human designer. We present a drama manager that learns a model of the player's storytelling preferences and automatically recommends a narrative experience that is predicted to optimize the player's experience while conforming to the human designer's storytelling intentions. Our drama manager is also capable of manipulating the space of narrative trajectories such that the player is more likely to make choices that result in the recommended experience. Our drama manager uses a novel algorithm, called Prefix-Based Collaborative Filtering (PBCF), that solves the sequential recommendation problem to find a sequence of plot points that maximizes the player's rating of his or her experience. We evaluate our drama manager in an interactive storytelling environment based on choose your own adventure novels. Our experiments show that our algorithms can improve the player's experience over the designer's storytelling intentions alone and can deliver more personalized experiences than other interactive narrative systems while preserving players' agency.**

***Index Terms*—Interactive story generation, player modeling, drama manager, prefix-based collaborative filtering**

I. Introduction

An *interactive narrative* is a form of digital entertainment in which users create or influence a dramatic storyline through actions, typically by assuming the role of a character in a fictional virtual world. The primary goal of an interactive narrative is to make players feel like they are part of a coherent unfolding story while also affording them the opportunity to act in a way that can change the direction or outcome of the story. There are many ways to achieve interactive narrative. A common technique that does not require artificial intelligence is to construct a *branching story graph*, a directed acyclic graph in which nodes contain narrative content (e.g., plot points) and arcs denote alternative choices of action that the player can choose. Branching story graphs are found in the choose your own adventure series of novels [1], and also used to great effect in hypermedia and interactive systems [2].

Other approaches to interactive narrative employ a *Drama Manager* (DM), an omniscient background agent that monitors the fictional world and determines what will happen next in the player's story experience. The goal of a drama manager is to coordinate and/or instruct virtual characters in response to player actions, thus achieving a coherent narrative experience while preserving the player's agency in the world [3], [4]. In other words, a drama manager attempts to enhance the player's experience in the virtual world by manipulating the player's progression through a space of story content. To achieve this goal, a drama manager must generate and evaluate possible future narrative sequences that the player could experience.

In the prevailing approaches to drama management, a human game designer provides a global set of goals and heuristics that determines what a "good" story should be. This high-level designer specification is often the only measure of quality of the player's interactive experience. In short, *the Drama Manager is a surrogate for the human designer*. The drama manager uses this information to increase the chances that the player will encounter a sequence of plot points that will be rated highly according to this designer-specified definition of good experience [4]–[10].

We believe that different players have different preferences over the game stories they would like to experience and that a drama manager should take the players' preferences into consideration when determining how the narrative should unfold. To that end, *we argue that a drama manager must* **also** *be a surrogate for the game players*. We aim to create a drama manager that can deliver personalized stories that satisfy both the designer's intentions and player preferences. Personalized drama management requires two challenges to be met:

1) The drama manager must learn a player model that can be used to predict players' preferences over possible future narrative sequences.
2) The drama manager must manipulate the space of narrative trajectories such that the player is more likely to make choices that result in a preferred experience.

Player modeling in games can be treated as a content recommendation problem [11]. We convert the player modeling problem into a collaborative filtering (CF) problem. Collaborative filtering has been successfully applied in recommender systems to model user preference over movies, books, music, and other products [12]. CF algorithms attempt to learn users' preference patterns from ratings feedback and predict new user's ratings from previous user's ratings which share similar preference patterns. A drama manager agent must continuously ask the question "what plot point should happen next given designer constraints and player preferences?" In this work, we extend CF algorithms to recommend *sequences* of plot points for the purpose of generating personalized story experiences. Unlike the traditional use of CF for *one-shot* recommendations of complete story artifacts (e.g., books, movies), we describe a new type of recommendation problem called *sequential recommendation* in which each subsequent recommendation is dependent on the entire sequence of prior recommendations for a particular story experience.

A drama manager utilizing a player model can optimize an individual player's narrative experience by selecting the branches through a branching story graph that are predicted to earn the highest ratings. However, this comes at the expense of *player agency*—the ability for the player to determine his or her own narrative future. Unfortunately, players are necessarily not aware of the future narrative consequences of their choices. We present an approach to personalized drama management whereby the drama manager manipulates the branching story graph such that the player is more likely to make a choice that coincides with the optimal trajectory through the branching story graph. Given multiple user options that lead to the same plot points, our drama manager uses collaborative filtering to predict which option a player is likely to choose and only presents a subset of options such that the player is more likely to follow the branch that the DM desires.

This article presents two novel contributions to the area of personalized drama management. First, we introduce a novel solution to the sequential recommendation problem called *Prefix-Based Collaborative Filtering* (PBCF), which learns a player's preferences over fragments of stories and chooses successive plot points to generate a story experience that optimizes the player's enjoyment. A DM built with the PBCF algorithm is capable of learning a robust player model to generate personalized stories that are built directly from players' structured feedback, i.e. ratings, without rigid player type presumptions. Furthermore, our PBCF based drama manager can learn the player's preference incrementally; as the player leaves more feedback, the drama manager will build a better player model based on all previous ratings and predict the successive plot point with increasing accuracy. An evaluation of the PBCF algorithm has been performed on a controlled testbed domain based on Choose-Your-Own-Adventure book serials using human players and simulated players. Second, we present a drama manager that can manipulate players into making optimal narrative choices without reducing player agency. A preliminary study on human players shows that our drama manager can significantly alter the narrative trajectories of players.

II. Related Work

Drama manager agents have been widely used to guide the players through an expected story experience set by designers. Two approaches to drama management—search-based drama management [4], [5], [13] and declarative optimization-based drama management [6], [14]—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 [9] uses a reactive plot point selection technique to determine the next set of behaviors for two virtual characters. Riedl et al. [7] use a partial-order planner to re-plan a story when the player performs actions that change the virtual world in ways that prevent story progression as expected. Similarly, Porteous and Cavazza [15] use a planner with designer-provided constraints to control virtual characters and push a story forward. A DM by Magerko et al. [8] 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.

Player modeling has been widely applied to adapt computer games [16]. But relatively little work has been done to use player models to determine how a story should unfold in a game or virtual environment. The PaSSAGE system [17] 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 CYOA style story graph. PaSSAGE uses Robin's Laws five game player schemes: Fighters, Power Gamers, Tacticians, Storytellers, 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 [18] use the same player types. Seif El-Nasr [19] 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 player 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 and that plot points can 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 player feedback—ratings—and further solves sequential plot point recommendation/selection problems.

Roberts, et al. [6], [14] develop a drama manager 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 is 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 player. While the simulated player 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.

III. Branching Stories

We make the assumption that stories can be decomposed into a sequence of plot points, chunks of story content encapsulating character interactions, dialogue, or other noteworthy events. To recommend a story that is expected to maximize players' enjoyment in an interactive story-based game or virtual world, a drama manager needs to choose a subset of plot points and order them in an optimal sequence, which is an NP-hard problem given all possible plot points. To make the problem tractable, temporal and semantic constraints

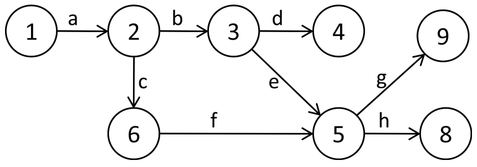


Fig. 1. A branching story graph illustrating a example of a simple story library.

are imposed among these plot points to reduce the size of the space of possible interactive story experiences [4], [5]. When constraints are known, a *branching story graph* can be derived automatically or manually that specifies the possible successors to each plot point. Figure 1 shows a simple branching story graph; each node is a plot point (numbered) and arcs denote choices (lower-case letters) that players can make after each plot point. A path through the graph starting from the root node and terminating at any leaf node is a possible complete story experience for the player. Figure 1 contains five possible complete stories: ({1,2,3,4}, {1,2,3,5,8}, {1,2,3,5,9}, {1,2,6,5,8}, and {1,2,6,5,9}). The branching story graph provides the DM a set of options for the next plot point at each node.

Although the concept of a branching story graph is simple, many other plot point representations used by AI-driven interactive narrative systems are reducible to the branching story graph [2]. This is due to the fact that a drama manager must generate possible future narrative experiences in order to select the best next plot point. In this work, we assume there exists a pre-authored branching story graph, which is stored in our *story library*. The branching story graph may have been authored by hand or by some other intelligent process (c.f., [2], [4]–[6]) or through collaborative editing techniques such as crowdsourcing [20]. This assumption allows us to focus on the DM decision-making process and the development and validation of our player modeling technique.

In the following two sections, we will introduce our prefix-based CF algorithm to recommend the next best plot point based on a player model's prediction of the enjoyment of the subsequent narrative experience and our evaluation results. In Section VI and VII, we will describe our algorithm to guide players to the recommended plot points and our preliminary experiment results.

IV. Prefix-Based Collaborative Filtering

We aim to bring a personalized story experience to players with different preferences. To recommend stories the players enjoy, we use collaborative filtering to build a *player model* that captures players' preferences and is capable of extracting and predicting the preferences of individual players over future narrative experiences. In this work, we extend CF algorithms to choose the best next plot points in the branching story graph.

To deal with the sequential nature of recommending a story plot-point by plot-point, we cannot simply adopt a CF algorithm and make one-shot recommendations of the best next plot point. In other words, drama management is a *non-Markovian* problem—at each step the DM's selection of best next plot point is based on all previous plot points encountered by the player. For example, in Figure 1 if the player is currently at node 3, the DM's next selection, node 4 or node 5, should depend on the player's preference on {1, 2, 3, 4}, {1, 2, 3, 5, 8}, {1, 2, 3, 5, 9}, instead of preference on the individual node 4 or node 5. The preference on {1, 2, 3, 4}, {1, 2, 3, 5, 8}, {1, 2, 3, 5, 9} is likewise correlated with players' preference on previously experienced paths {1}, {1, 2}, {1, 2, 3}, etc.

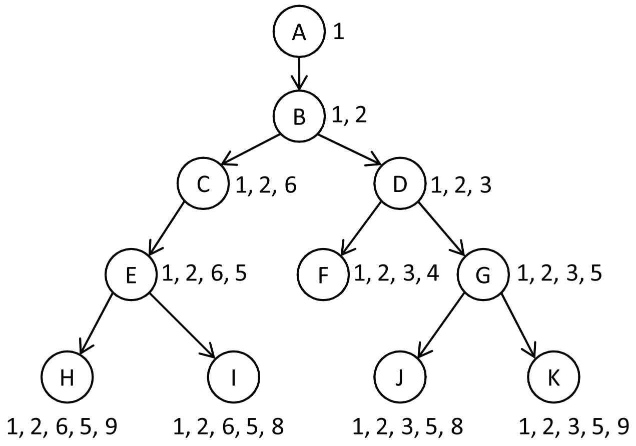


Fig. 2. The corresponding prefix tree of the branching story graph in Figure 1. The branching story graph is usually a graph while the prefix tree is a tree or forest.

In addition, we would like to avoid making any strong assumptions about the dimensions—the player types—of the player model. Previous approaches to personalization of interactive stories [17], [18], [21] require a designer to pre-determine the meaningful player types, even though there is no comprehensive theory that these pre-defined player types can cover all different players or can be generalized to different types of games, nor any clear evidence of links between player type models and preferences for story content. We believe the player types should be discovered directly from players' preferences and CF algorithms enable us to easily extract the player types by observations of players' structured feedback, i.e. ratings.

A. *Prefix Tree Representation*

The first step to addressing the sequential recommendation problem is to transform the branching story graph into a *prefix tree* as in Figure 2, where every node (upper-case letters) is a prefix of a possible complete story (i.e., a path from initial node to terminal node in a branching story graph). The children of a node in the prefix tree are the prefixes that can directly follow the parent prefix. Comparing Figures 1 and 2, one can see that each node in the prefix tree incorporates all the previous plot points in the path from the initial plot point to the current plot point.[1] With the prefix tree, the drama manager does not need to worry about history when recommending the next node. In our system, only the prefix tree is stored in the story library.

[1]All plot points from an original branching story graphs and all prefixes in a prefix tree are assigned unique identifiers, enabling the system to translate back and forth between representations as necessary.

In our approach, stories are presented to the players plot point by plot point and we collect players' ratings for 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 players to rate the story-so-far than the new plot point only since the history of interaction up to that point matters in stories; any one plot point does not make sense without previous ones. The story-so-far exactly corresponds to the nodes in the prefix tree. By converting a branching story graph to a prefix tree we do not need to solve the credit assignment problem as in reinforcement learning—how much of a final rating each previous plot point is responsible for [22].

Compared to other algorithms which also roll history into state nodes such as TTD-MDP [6], [14], our prefix-based collaborative filtering algorithm focuses on optimizing the path for different players based on the player model. Furthermore, unlike TTD-MDP which tries to convert the problem into a Markovian problem, the prefix selection problem in the paper is still non-Markovian. For example, if the DM is at node *D* in Figure 2, the selection of the next node (*F* or *G* in Figure 2 should be related to the player's ratings on previous three prefix nodes (*A*, *B*, and *D*). A player 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*. Through the prefix-based CF algorithm we describe in the next sections, the DM can model players' preference based on all previous prefix ratings and make a recommendation to the best of its knowledge.

B. Player Modeling

With the prefix tree, a *prefix-based* collaborative filtering algorithm can now be considered to build the player preference model. In prefix-based collaborative filtering, all the players' ratings for the story prefixes are collected in a single matrix which we call a *prefix-rating matrix*. An n by m prefix-rating matrix contains the ratings for n prefixes from m players. Each column of the matrix represents the ratings of the corresponding player for all the prefixes. Each row of the matrix represents ratings for the corresponding prefix from all the players. Figure 3 shows a simple illustration of the prefix-rating matrix. The matrix is usually very sparse, i.e. containing a lot of missing ratings which are labeled as $*$ in the figure, because we do not expect any given player to have read and rated all the prefixes in the library. Note that a single prefix-rating table contains entries for prefixes in all prefix trees in the forest making up the story library.

If we can predict all the missing ratings in the prefix-rating matrix for a player, it will be straightforward to recommend the best next prefix during story recommendation process—to pick the prefix that will lead to the highest rated stories. In our approach, the prefix-rating matrix is treated as the product-rating matrix as in traditional collaborative filtering [12], [23]. CF algorithms can be applied to train and compute the missing ratings in the prefix-rating matrix.

Our CF algorithms make no presumptions on the player types. Instead the algorithms will cluster the ratings of the players and learn "patterns" from each cluster of ratings. These "patterns" also represent player types, although, as with many Machine Learning techniques, it is difficult for us to interpret these player types. These player types are soft clusters in the sense that a particular player can have a degree of membership in each player type. The learned player types are more capable of describing all types of players for particular games, compared to the pre-defined types. Next we will introduce the CF algorithms used in the paper, followed by the description of two phases of our plot point sequence selection process: model learning and story recommendation.

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	*	...
...	...	...	...	...

Fig. 3. An illustration of the prefix-rating matrix. *A*, *B*, *C* and *D* represent the prefixes in Figure 2. The larger the digital number, the higher the preference. The stars represent those missing ratings.

C. Collaborative Filtering Algorithms

We have experimented with two collaborative filtering algorithms: *probabilistic Principle Component Analysis* (pPCA) [24] and *Non-negative Matrix Factorization* (NMF) [25], [26]. We will briefly introduce the two algorithms and their application to our model learning and story recommendation process.

The probabilistic PCA algorithm assumes that a n dimensional vector $\mathbf{r}$ can be factorized as follows:

$$\mathbf{r} = W\mathbf{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{1}$$

where $\mathbf{x}$ is a n' dimensional vector in the hidden or reduced dimension space (usually $n' < n$) and $\mathbf{x} \sim N(0, \mathrm{I})$. W is a n by n' matrix. $\boldsymbol{\mu}$ is the mean vector which permits $\mathbf{r}$ to have nonzero mean. $\boldsymbol{\epsilon} \sim N(0, \sigma^2\mathrm{I})$ is the Gaussian noise.

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

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

Thus the basic assumption of pPCA algorithm is that the player's prefix rating vector (the column of the prefix-rating matrix) obeys a multi-dimensional Gaussian distribution. In other words, the ratings for each prefix from a single player take a univariate Gaussian distribution. Furthermore, pPCA assumes that the expectations of different players' rating vectors are linear combinations of w_i, the columns of the matrix W, which in our case represent player types. The player model in pPCA is captured by W, μ and σ. Thus, for each player with prefix-rating vector $\mathbf{r}_i$, these parameters help us find the hidden vector $\mathbf{x}_i$, the individual player's preference properties;

once the hidden vector is known, the player's ratings for all prefixes without ratings by this individual can be computed according to the multi-dimensional Gaussian distribution.

The NMF algorithm aims to factorize an n by m matrix R as follows:

$$R = W * H \tag{3}$$

where $W \in \mathbb{R}^{n*n'}$ and $H \in \mathbb{R}^{n'*m}$ are two non-negative matrices (usually $n' < n$). The non-negative property means that all the entries in the matrix are greater than or equal to zero.

In our case, R is set to be the prefix-rating matrix (n prefixes and m players). The player model in NMF is simply the matrix W. The columns of the matrix W, $\boldsymbol{w}^j$ $j = 1, ...n'$, are bases that represent different types of players. $\boldsymbol{h}^i$, the i^{th} column of H, corresponds to the i^{th} player's preference properties.

In practice, it will be difficult to interpret the player types that correspond to $\boldsymbol{w}^j$ or $\boldsymbol{h}^i$. However, known player types can be introduced as prior knowledge to the model learning process to improve the training accuracy. For example, in NMF, if we have prior knowledge about some preference types (e.g., fighter, tactician), i.e., 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 V-F show that such prior knowledge can indeed increase player modeling accuracy when they are known to accurately distinguish players with regard to stories. More details about how the CF algorithms are applied in our system are explained in the following sections.

D. Model Learning Algorithms

Due to the large amount of missing values in the prefix-rating matrix R, the EM algorithm [27] is used to learn the parameters for pPCA algorithm (W, $\boldsymbol{\mu}$ and σ) and NMF algorithm (W).

1) Model Learning with pPCA: In the E-step of the pPCA model learning algorithm, we use a Gaussian Process to compute the missing ratings in R given the parameters W, $\boldsymbol{\mu}$ and σ [28]. Let $\Sigma = WW^T + \sigma^2 I$ be the covariance matrix for the rating vector $\mathbf{r}$, which is one column of R. Denote the sub-vector of $\mathbf{r}$ which contains all missing ratings as $\mathbf{r}_h$ and the sub-vector of $\mathbf{r}$ which contains all known ratings as $\mathbf{r}_o$. Then the distribution $p(\mathbf{r}_h|\mathbf{r}_o)$ and the expectation of $\mathbf{r}_h$ can be computed using the Gaussian Process:

$$\mathrm{E}(\mathbf{r}_h|\mathbf{r}_o, \boldsymbol{\mu}, \Sigma) = \boldsymbol{\mu}_h + \Sigma_{ho}(\Sigma_{oo})^{-1}(\mathbf{r}_o - \boldsymbol{\mu}_o) \tag{4}$$

where $\boldsymbol{\mu}_h$ is the sub-vector of $\boldsymbol{\mu}$ containing elements at the positions corresponding to the missing ratings and $\boldsymbol{\mu}_o$ is the sub-vector containing elements at the positions corresponding to the known ratings. Σ_{ho} means the sub-matrix of Σ, of which the rows are indexed by the position of missing ratings while the columns are indexed by known ratings in $\mathbf{r}$. The notation system follows the tradition in [23].

In the M-step of the pPCA, the parameters W, $\boldsymbol{\mu}$ and σ are computed through maximizing the expected likelihood function $\mathrm{E}(\mathbf{r}_h, \mathbf{r}_o|W, \boldsymbol{\mu}, \sigma)$ over distribution $p(\mathbf{r}_h|\mathbf{r}_o)$, which is computed in the E-step using Gaussian Process. After a few equation manipulations, the expected likelihood function will be:

```
1: Initialize $W$, $\boldsymbol{\mu}$ and $\sigma$ randomly for pPCA, or $W$ and $H$ for NMF
2: while not converging or termination criterion not reached do
3:     Compute the rating matrix $R$ through Equation 4 for pPCA, or Equation 3 for NMF {E-step}
4:     Set the corresponding elements in $R$ to the known ratings in $R_0$
5:     Compute $W$, $\boldsymbol{\mu}$ and $\sigma$ through Equation 6-8 for pPCA, or $W$ and $H$ through Equation 9 and Equation 10 for NMF {M-step}
6: end while
```

Fig. 4. The model learning algorithm.

$$\mathrm{E}(\mathbf{r}_h, \mathbf{r}_o|W, \boldsymbol{\mu}, \sigma) \sim \log|\Sigma| + \mathrm{tr}(C\Sigma^{-1}) \tag{5}$$

where $C = \frac{1}{m}\mathrm{E}(\sum_{i=1}^{m}(\mathbf{r}_i - \boldsymbol{\mu})(\mathbf{r}_i - \boldsymbol{\mu})^T)$ and m is the total number of training players. The parameters can be computed by minimizing Equation 5. The minimization results are as follows:

$$\boldsymbol{\mu} = \frac{1}{m}\sum_i \mathbf{r}_i \tag{6}$$

$$\sigma^2 = \frac{\mathrm{tr}(C) - \sum_{i=1}^{n'}\lambda_i}{n - n'} \tag{7}$$

$$W = U'S \tag{8}$$

where the λ_i is the ith biggest eigenvalue of C, U' contains the n' eigenvectors corresponding to λ_i and S is a diagonal matrix with the ith value equaling to $(\lambda_i - \sigma^2)$.

2) Model Learning with NMF: For the NMF algorithm, the E-step to compute the prefix-rating matrix R is simple given the W and H: $R = WH$. Then the known elements of R are set to be the corresponding input player ratings.

Given a fully observed R in the M-step, the objective is to minimize the distance $||R - WH||^2$ as in [25] to get W and H, where $|| \circ ||$ is the Frobenius norm. The update rules are as follows:

$$H_{ij} \leftarrow H_{ij}\frac{(W^T R)_{ij}}{(W^T W H)_{ij}} \tag{9}$$

$$W_{ij} \leftarrow W_{ij}\frac{(RH^T)_{ij}}{(WHH^T)_{ij}} \tag{10}$$

3) Summary of Model Learning: Regardless of algorithm, the complete model learning process is as follows. First we build the story library with the prefix tree. Second, we collect data to populate the prefix-rating matrix R_0. Finally, we use the algorithm in Figure 4 to learn the player model, i.e., to predict missing rating values for all players.

E. Story Recommendation Algorithms

Story recommendation occurs once we have learned the model parameters using pPCA or NMF and subsequently wish to select a path through the branching story graph for an individual player. The story recommendation phase begins with collecting some initial ratings to seed $\boldsymbol{r}_0$ for a new player. Next, we predict the missing ratings for the player. When using pPCA, this is accomplished simply by applying Equation 4. For NMF algorithm, the prediction algorithm in Fig. 5 is used.

After we compute the vector $\boldsymbol{r}$ with no missing ratings, a valid next prefix that can lead to the highest rated full-length

```
1: Initialize $h$ for NMF
2: while Not converging or termination criterion not reaching do
3:     Compute $\boldsymbol{r}$ using Equation 3 (with $h$)
4:     Set the corresponding elements in $\boldsymbol{r}$ to the known ratings in $\boldsymbol{r}_0$
5:     Compute new $h$ using Equation 9
6: end while
```

Fig. 5. The rating prediction algorithm for NMF.

story will be selected. For example if the story has proceeded to node B in Figure 2, the selection of node C or D depends on the predicted ratings of node H, I, J and K. If node I gets the highest predicted rating, then node C in the graph will be selected. If node K wins, node D will be selected.

V. Evaluation of Prefix-Based Collaborative Filtering

In this section, we temporarily set aside the question of guiding players to the recommended story plot points and focus on evaluation of the quality of the stories recommended by our prefix-based CF algorithm. We have built a simplified storytelling system in which the drama manager fully controls the story progression using the PBCF algorithm. Although the player agency is completely removed, the simplified storytelling system facilitates us to examine our PBCF algorithm under controlled conditions. Three human studies have been performed to evaluate the player enjoyment of the recommended personalized stories.

The first experiment we conducted trained a player model on human ratings of stories. We evaluate pPCA and NMF implementations for the accuracy of models of human rating behavior learned. The second experiment uses the player model trained in the first experiment to evaluate the story recommendation algorithm against a baseline. Finally, we ablate our system to assess whether the plot point recommendation problem can be solved with a more classical non-sequential recommendation approach. We also performed experiments on simulated players to further evaluate the performance of different story/prefix recommendation algorithms.

A. Story Library

The story library we used for our experiments was built by transcribing the stories from four Choose-Your-Own-Adventure (CYOA) books: *The Abominable Snowman*, *Journey Under the Sea*, *Space and Beyond*, and *The Lost Jewels of Nabooti*, all of which contain adventure stories for teenagers in the US [1]. The stories from one book constitute a branching story graph. At the end of every page in the book, there is a multi-choice question. Depending on which choice the reader chooses, he or she will be delivered to different pages of the book to continue down different branches of the story. Figure 6 shows a branching story graph from one of the books.

One of the reasons that we transcribed stories from Choose-Your-Own-Adventure books is to control for story quality, as opposed to authoring stories ourselves. In the current system, every story was pruned and transcribed to contain exactly six plot points. We did this only for implementation purpose,

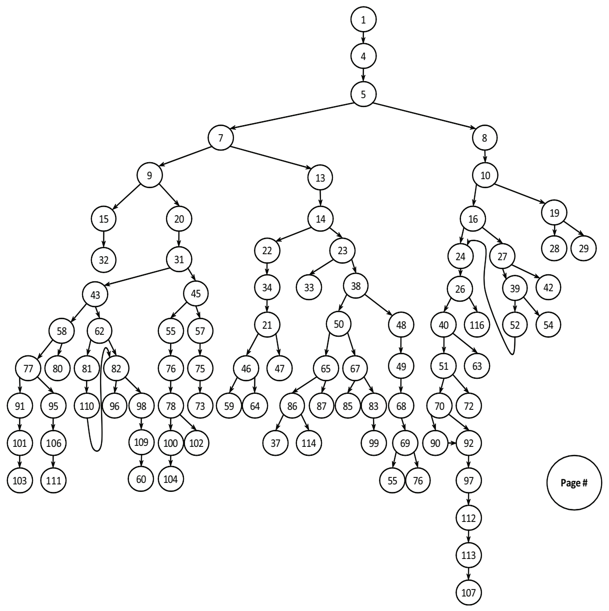


Fig. 6. Illustration of the branching story graph of stories in *The Abominable Snowman*. Each node in the graph represents a page in the book (a plot point). Every story starts from the root node and ends on one of the leaves.

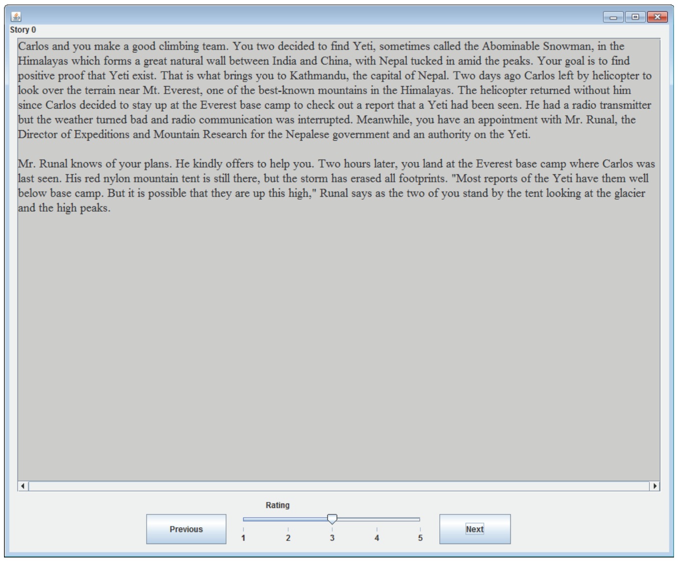
although our algorithm generalizes to longer and variable-length stories. We further removed branches that are shorter than six plot points due to the sudden demise of the player's character. The four branching story graphs were transformed into a forest of four prefix trees,[2] which was stored in the story library. In total, our story library is capable of generating 154 possible stories (on average 1000 words per story) and contains 326 prefixes. That is, there are 154 paths through the four branching story graphs from the four Choose-Your-Own-Adventure books and thus 154 leaf nodes in the prefix forest in the library. The constraints between plot points greatly reduce the number of valid stories in the story library; the number of prefixes will grow linearly with the number of full-length stories.

B. User Interface

We built a simple storytelling system to examine our prefix-based CF algorithm. Our system implements the CF model learning and story recommendation algorithms described earlier. We have disabled the ability for players to make choices in order to assess the story recommendation capabilities of our system. The drama manager recommends plot points and directly presents them to the players. It is thus capable of guiding a player through a particular path from initial plot point to a terminal plot point that is believed to be most enjoyed by the player.

Figure 7 shows a screenshot of the simple storytelling system we built for experiments with human participants. The system presents the stories to the player one plot point at a time, where each plot point is a paragraph of a story (Figure 7 shows two plot points). After each plot point, the system

[2]Due to the source of our data set, the trees in our prefix forest are disjoint, although this does not always need to be the case.



Carlos and you make a good climbing team. You two decided to find Yeti, sometimes called the Abominable Snowman, in the Himalayas which forms a great natural wall between India and China, with Nepal tucked in amid the peaks. Your goal is to find positive proof that Yeti exist. That is what brings you to Kathmandu, the capital of Nepal. Two days ago Carlos left by helicopter to look over the terrain near Mt. Everest, one of the best-known mountains in the Himalayas. The helicopter returned without him since Carlos decided to stay up at the Everest base camp to check out a report that a Yeti had been seen. He had a radio transmitter but the weather turned bad and radio communication was interrupted. Meanwhile, you have an appointment with Mr. Runal, the Director of Expeditions and Mountain Research for the Nepalese government and an authority on the Yeti.

Mr. Runal knows of your plans. He kindly offers to help you. Two hours later, you land at the Everest base camp where Carlos was last seen. His red nylon mountain tent is still there, but the storm has erased all footprints. "Most reports of the Yeti have them well below base camp. But it is possible that they are up this high," Runal says as the two of you stand by the tent looking at the glacier and the high peaks.

Fig. 7. A screenshot of our simplified storytelling system. A plot point is a paragraph of the story in the figure.

asks the player for their preference rating on the story-so-far (corresponding to a prefix node in the prefix tree). The ratings are integers ranging from 1 to 5 where a larger number indicates a higher preference. A new plot point will appear after the player clicks the *Next* button. The next plot point is determined by the story recommendation algorithm, which is either PBCF or random selection of a successor.

By temporarily 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 player model. An interactive storytelling system that gives players the appearance of full agency will be described in Section VI.

C. Experiment 1: Training the Player Model on Human Players

In this experiment, we examine the ability of our system to learn a player model utilizing human-generated prefix ratings. To learn a model, we need a sparse prefix-rating matrix. To generate the matrix, we implemented a version of our storytelling system that randomly walks the branching story graph. It starts at a randomly selected root node and then selects randomly from possible successor plot points. For each plot point presented, the system asks the player for a rating of the story-so-far.

We recruited 31 players (18 male and 13 female) for the experiment. 26 of the players are college graduate students and the other 5 players are research scientists and staff at our University. Five out of the 31 players were familiar with the choose your own adventure series of books prior to participating in the study. Participants who were not familiar with choose your own adventure books were given a sample adventure story to familiarize them. The experiment took about half an hour for each player.

We obtained a 326 by 31 prefix-rating matrix R with $\sim 86\%$ ratings missing. The prefix-rating matrix R was 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. The R^t and the R^v are still of the same dimensions as the original R and they both contained missing values.

TABLE I
THE AVERAGE RMSE FOR NMF AND PPCA ALGORITHMS WITH DIFFERENT PARAMETERS.

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

We trained the NMF and pPCA algorithms on the training set R^t with different parameters. The resulting models were then used to predict the ratings in the validation matrix. To evaluate the accuracy of each algorithm, we measured the Root Mean Square Error (RMSE), which is computed as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{|O|}\sum_{i,j\in O}((R^v)_{ij} - (R^{v'})_{ij})^2} \qquad (11)$$

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 ten times and the average RMSEs on the validation sets are reported in Table I. The dim$i$ in the table means that from NMF the matrix W in Equation 3 has i columns. The RMSEs in the table suggest that there are probably six types of players in the current training set when it comes to story preferences. Although we cannot easily interpret and label the player types, by learning to cluster participants into these player types the system is able to predict prefix ratings to within one point of actual participant ratings on the story-so-far on average.

D. Experiment 2: Evaluation of Recommended Stories

We wish to confirm the hypothesis that the PBCF algorithm can improve overall ratings of story experiences by increasing the likelihood that players see stories that are more aligned with their preferences than by chance alone. Because of constraints between plot points, we know that all possible paths in a branching story graph from root to leaf nodes were intentionally designed, even random walks produce good stories from the designer's perspective. Our hypothesis builds on the assumption that individual players may have preferences across the possible story experiences allowable by the designer's branching story graph.

22 graduate students (17 male and 5 female) were recruited to evaluate our PBCF algorithm. None of the participants were involved in earlier experiments. We use the best player model from Experiment 1 (i.e., NMF with 6 dimensions). To compare the model performance on new players versus existing players

TABLE II
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 HIGHER THAN THE AVERAGE RATING OF THE RANDOM STORIES.

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

on which the player model was trained, we also invited 11 players from experiment 1 back to participate in this study.

The story recommendation study consists of four stages. In the first stage the DM presents five randomly generated stories, generated in the same way as the first training experiment. This provides some initial ratings from the participant. In the second stage, five personalized stories are recommended according to our PBCF algorithm. These personalized stories are also presented to the participant plot point by plot point and the players' story-so-far ratings are collected after every plot point. As in Sharma et al. [13], the DM presents another five personalized stories in the third stage, followed by five random stories in the last stage in order to eliminate any bias introduced by the order in which the stories are presented to the participants. In total, every participant is required to read 20 stories (10 total random stories and 10 total personalized stories).

Table II shows the results for the new players and returning players. The first line exhibits the statistical results on all the 33 testing players. The second and the third lines give the results of the 11 returning players and the 22 new players, respectively. The first column "Random" and the second column "Personalized" show the average ratings of all the random and all the personalized stories respectively. For every player in the story recommendation 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 chooses preferred stories. The last column shows the significance of the difference between random and personalized averages using a one-tailed t-test.

One reason we would like to collect players' ratings during story recommendation phase is to create a more accurate player model as described in Section IV-E. Another advantage is that both the random stories and personalized stories will be presented in the same format so that the players' preference judgement will not be affected by the system interface. In fact, the players were not told which stories were random and which stories were personalized before the experiment and most of them did not notice the difference afterwards. The last reason is that we need to compare their ratings between random stories and personalized stories so that we can evaluate the story recommendation algorithm.

TABLE III
THE EXPERIMENT RESULTS OF THE COMPARISON EXPERIMENT WITHOUT CONSIDERING HISTORY.

	Random	Personalized	Accuracy	p-value
All	2.4214	2.5500	0.464	0.5621

E. Experiment 3: Sequential versus Non-Sequential Plot Point Recommendation

Throughout the paper, we assume that the order in which one experiences plot points affects players' story ratings. However, it is possible that the only thing that matters is the most recently encountered plot point. We test this assumption by attempting to learn a player model that recommends plot points without considering sequence.

The comparison experiment is designed in which the DM recommends the successive plot points based on the players' ratings over plot points. This study was also composed of model training phase and story recommendation phase. In the model learning phase, 19 players participated in the study. Each player read 30 independent plot points without story context. To control for participant perception of a temporal connection between plot points, plot points were randomly selected and any two consecutive plot points were always chosen from different CYOA books. We additionally altered the user interface so that only one plot point is visible at a time. Participants were told to rate single plot points irrespective of prior plot points. Instead of the prefix-rating matrix, we collected a plot point rating matrix in the model learning phase. The player model was built by training the NMF algorithm (with six dimensions) on the plot point rating matrix.

In the story recommendation phase, we invited another 14 participants for the study. The game interface is similar to that used in prior experiments, but only showing one plot point on the screen at a time. Every player read 20 stories in exactly the same sequence as in previous experiments (5 random, 5 personalized, 5 personalized and 5 random). And after each plot point, the participant was required to give a preference rating for the plot point. Based on the NMF model learned from the plot point rating matrix, the DM chose the next plot point that could follow the current plot point to guide the players through the branching story graph.

Note that the players' ratings are still dependent on the story-so-far since it is impossible for them to forget previous plot points. As a result, in this experiment, the drama manager tries to predict players' preference on story prefixes using the player model built based on players' preference for plot points. Table III shows the results of the comparison between the system's attempt to generate personalized stories and random choices. There is no significant difference between the average rating of personalized stories and random stories, suggesting that history matters when recommending plot points.

F. Experiment 4: Evaluation of Story Recommendation with Simulated Players

In addition to studies with human participants, we also conducted experiments on simulated computer players in order to get a more complete picture of PBCF algorithm performance. Simulated players are more consistent over time, allowing us to make observations about our algorithms on a controllable data set. With simulated players we can generate larger data sets to examine algorithm learning rates and experiment with different algorithm designs without requiring hundreds of human participants. Note that we are not requiring that the simulated players play in the same way or have the similar preference as human players. Instead, as long as the simulated players are consistent on their behaviors, they can be used to test the capability of our system to capture players' preference and build player models.

The simulated players are built based on the Robin's Laws player schemes and used in related works (cf., [17], [18]). Every simulated player is created with a five-dimensional characteristic vector. Each entry of the vector (ranging from 0 to 1) specifies the corresponding characteristic of the simulated player. For example, the vector $[0, 0.7, 0, 0, 0.9]$ means the simulated player is a combination of Power Gamer and Method Actor and tends to enjoy Method Actor a little more.

Story prefixes are labeled according to beliefs about how they match Robin's Laws player schemes. Note that the prefix labels are not required to be accurate descriptions of the story prefix content since we do not aim to imitate human preference.

Furthermore, we assume that a simulated player will prefer a story prefix that most closely matches the player's type. For example, a simulated player with characteristic vector $\mathbf{p} = [0.8, 0, 0, 0, 0]$ will prefer for a story prefix i with label $\mathbf{s}_i = [1, 0, 0, 0, 0]$ over a story prefix j with label $\mathbf{s}_j = [0, 1, 0, 0, 0]$. Consequently, we assume that the rating r of a simulated player $\mathbf{p}$ for a prefix $\mathbf{s}$ is proportional to cosine distance between the vector $\mathbf{p}$ and $\mathbf{s}$: $r \sim \frac{\mathbf{p}^T\mathbf{s}}{|\mathbf{p}||\mathbf{s}|}$. The ratings are computed by scaling the cosine distances to the range 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 variability in the human player behaviors that it could be inaccurate to quantitate preference into digital labels.

In the model learning phase, 120 simulated players were created 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 player then "read" 10 random stories and generated a preference rating after every plot point, similar to what we asked of human participants. A 326 by 120 prefix-rating matrix was generated in this way and used to train the player model.

In order to test the generalization ability of our PBCF algorithm, a new group of 1000 simulated players was created in the story recommendation phase. Each simulated player was given a random characteristic vector, of which the five entries were floating point values ranging from 0 to 1. The story recommendation phase follows the same four steps as the human story recommendation experiments. For the purpose of comparison to other algorithms, the DM generated personalized stories using the following algorithms:

- *BaselineP*: using pPCA to learn the player models based only on simulated players' ratings for full-length stories instead of prefixes, then directly recommend the full-length stories instead of choosing branches through recommending prefixes. The BaselineP 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 as the CF algorithm.
- *Vector*: A vector based player modeling algorithm that is similar to the model learning technique used by Thue et al. [17]. Each player is simulated as a vector which initially is $[0, 0, 0, 0, 0]$. 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. Then the DM generates successive plot points by recommending the following prefix based on the updated player vector, or chooses randomly when there is no clear preference.
- *pPCA*: The prefix-based CF algorithm using pPCA; same as with the human players.
- *NMFwoP*: The prefix-based CF algorithm using NMF without prior knowledge; same as with the human participant story recommendation experiment in Section V-D.
- *NMFwP*: The prefix based algorithm using NMF with Robin's Laws player schemes as prior knowledge. In the case of simulated players, we can compute the accurate rating vector $\boldsymbol{w}^j$ for each known player type j, where $j = 1, ..., 5$ correspond to the five player types in the model learning phase. These vectors $\boldsymbol{w}^j$ are included in the matrix W in Equation 3 as fixed columns during the model learning process. This condition represents the near ideal circumstance where the designer has strong genre knowledge about how players respond to stories and can author plot point sequences accordingly.

TABLE IV
THE EXPERIMENT RESULTS FOR THE SIMULATED PLAYERS USING SEVERAL VARIATIONS OF THE DM ALGORITHM.

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

The experiment results for these algorithms on the 1000 simulated players are shown in Table IV. 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 simulated testing players.

We explored the learning rate of different player modeling algorithms. In previous experiments, each player read 5 stories (random or generated) in each of the four steps of the story recommendation phase. We alter this number and compare the story recommendation accuracy for different algorithms. Figure 8 shows the average accuracies of 1000 simulated players as the number of stories read in every step changes. As shown in the figure, the NMF algorithms can achieve

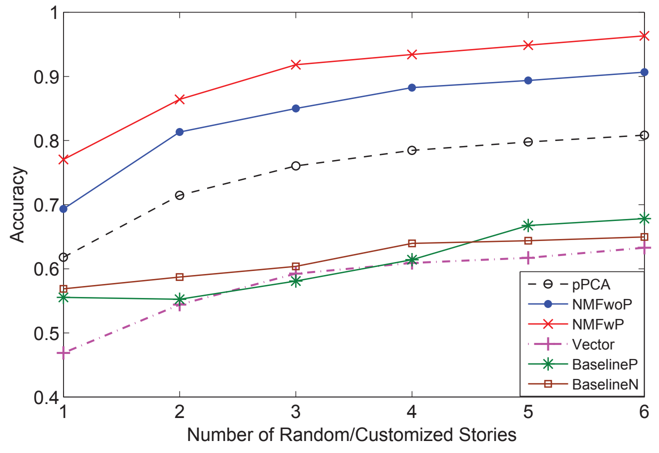


Fig. 8. The accuracies of the six algorithms as the number of stories read in every step changes.

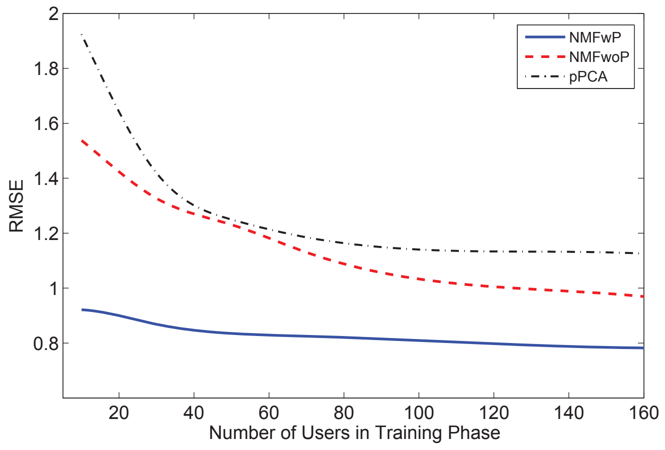


Fig. 9. The average RMSEs of the three prefix based algorithms with different number of simulated players for training.

accuracies higher than 70% even when a new simulated player reads only one story.

The influence of the training set size on the player model learning process was also tested. Figure 9 shows the average RMSEs of the three prefix based algorithms with different number of simulated players for training. Each RMSE value in the figure is an average computed from 10 random splits of the training data. As seen from the figure, the training RMSEs decrease as 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 players goes above 100 even if it has the perfect prior knowledge of the player models.

G. Discussion

The experiments on the human players using our prefix-based collaborative filtering algorithm achieve high story recommendation accuracies on the current Choose-Your-Own-Adventure data set. In the study, the new players rate DM-generated stories higher than random stories for over 80 percent of the time. Because of the expertly designed branching story graphs, even random stories are quality experiences; respecting player preferences increases the enjoyment. We achieve this accuracy rate after new players have only read and rated 5 sample stories. An accuracy of about 86% is achieved when the testing players' data are already part of the trained model. For all the participants including the returning players and the new players, the average ratings for the personalized stories are higher than the random stories, and the p-values are approaching zero for the results. The results show that our prefix based player modeling algorithms can capture the players' preferences and generate new stories with high accuracy.

We wondered if our assumptions about drama management should best be characterized as a sequential recommendation problem. When we prevented our algorithms from taking history into consideration and asked humans to rate plot points independently of each other, PBCF accuracy dropped to 56%, which is similar to random selections and the average rating of generated stories is similar to the average rating of random stories. This proves our assumption that a player model built on plot point preference cannot be used directly to predict players' preference for prefixes or full length stories. Thus the experiment proves the importance of history of experience in story recommendation, the theory based on which we build our prefix-based CF algorithm.

In the experiments with simulated players, the NMF algorithm usually performs better than the pPCA algorithm. One reason for it might be our linear model assumption for the simulated players. The linear characteristic model for simulated players coincides with the basic assumption of the NMF algorithm, which also assumes that players (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 simulated players, it is also superior to pPCA for human data in terms of RMSEs in our experiments.

Figure 8 illustrates that the prefix based algorithms are capable of extracting player preference much faster than traditional CF algorithms that create the player model directly from full-length stories (baseline algorithms *BaselineP* and *BaselineN*). This result demonstrates that the players' preference for story prefixes does correlate with players' preference for full-length stories. Compared to the experiments on human players without history in Section V-E, we can conclude that the player models built based on prefix ratings are better at describing the players' preference over full-length stories than the plot point based player model. Another reason the prefix based algorithms work better than traditional CF is that the prefix based algorithms can obtain more preference information (the ratings on all the prefixes) from the players in both the model learning and story recommendation phases. Figure 8 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 and must build its model of the player as the story unfolds.

One of the potential limitations to our approach is the need to train the player model on human players, requiring them to rate the story-so-far after every plot point. We acknowledge that this may disrupt immersion, resulting in inaccuracy in our player model. However, it is unlikely that a player will rate a non-preferred prefix higher than a preferred prefix just because the system has required too many rating interactions. That is, our data should be consistent. Once the prefix-rating matrix is sufficiently populated, our synthetic player studies suggest that PBCF requires as few as one story worth of ratings to achieve 70% prediction accuracy. Training is a necessary requirement if the player wishes a drama manager to work on one's behalf to increase the quality of one's narrative experience. After an initial, short training phase, we do not require the player to further rate the story-so-far, although by doing so the system can increase its prediction accuracy and thus ability to improve the player's experience. Thus, there is a trade-off between player willingness to provide data and quality of experience.

PBCF is scalable to large numbers of players and branching stories. The total number of prefixes in the story library is theoretically 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. 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 [12].

An analysis of the corpus of four choose your own adventure books reveals that 153 out of 154 of the possible trajectories through the branching story graphs (leaf nodes in the prefix tree) yield a mean rating of 2.5 or higher. Only one trajectories has a mean rating of 2.0, but the trajectory was only rated by two participants. The ratings of the set of possible full narratives appear to be normally distributed around a mean of 3.5. While some possible trajectories are better than others, the corpus analysis reveals that the books contain relatively good stories. That PBCF is able to improve players' experiences over random is noteworthy. That is, a personalized drama manager based on PBCF can take a well-authored branching story graph that would result in good narrative experiences without intervention and improve the subjective and individualized quality of experience.

VI. Personalized Drama Manager

In this section, we describe our approach to guiding players to the recommended plot points while giving the players the appearance of full agency. The challenge to be addressed is that players necessarily do not know how their choices at any given plot point will impact their future story experience. Thus, we hypothesize that players choose options based on a variety of local cues—those that sound most interesting, that agree with personal motivations, or that sound most likely to lead to favorable outcomes. Therefore, tension arises between the player and the drama manager when the player's choices do not coincide with the DM's desired narrative trajectory—the branch predicted to optimize the player's experience.

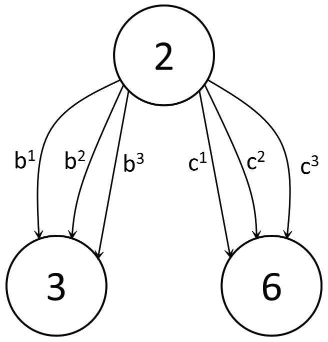


Fig. 10. Illustration of a portion of Figure 1. Multiple options point to the same plot point.

To address this challenge a drama manager must manipulate the branching story graph such that players are more likely to make choices in their own best interests *without* removing the player's ability to select any designer-specified branch. Our approach is to extend the branching story graph representation such that multiple options are allowed to point to the same child plot point. The extended representation can be seen in Figure 10, which is an enlargement of a single branch from Figure 1 such that b^1, b^2, b^3, c^1, c^2 and c^3 are options that can be presented to the player. The goal of the drama manager is to pick a subset of the options to present to the player such that at least one option leads to each child (ensuring true agency) and also increases the likelihood that the player will pick the option that transitions to the desired child plot point. For example, suppose the drama manager—using PBCF—predicts that the player's optimal narrative trajectory is through plot point 6 and further predicts that the player's preferences over options to be $b^1 > c^1 > b^2 > c^2 > b^3 > c^3$, such that the player is predicted to transition to plot point 3 instead. To intervene, the DM can present options c^1 and b^3 to the player, while suppressing the other options.

To guide a player to a recommended plot point without eliminating player agency, our approach is as follows. First, the drama manager uses PBCF to predict the best trajectory through the branching story graph for a given player, as detailed in Section IV. Second, the drama manager uses collaborative filtering to predict the player's preferences for options available to him or her at any given plot point. When the player is predicted to choose a branch that differs from the PBCF recommendation, the drama manager selects options to present or suppress. The player can still make a choice contrary to that recommended by PBCF, in which case the drama manager will attempt to guide the player down the next best narrative trajectory.

A. Player Modeling

We assume that players have different preferences for options. In order to predict players' option preference, we use collaborative filtering (CF) to learn players' preference

```
1: For each new player, collect an initial prefix rating vector $r_p$ and an
   option rating vector $r_o$, both of which contain missing values.
2: Present a root plot point to the player.
3: while the current plot point is not a leaf node in the branching story graph
   do
4:     Predict the missing ratings in $r_p$ using the algorithm in Section IV-E,
       and the missing ratings in $r_o$ using the algorithm in Section VI-A.
5:     Select one of the child plot point in the branching story tree that leads
       to the highest rated leaf node in the prefix tree.
6:     Given the desired child plot point, present two highest rated options
       that point to the desired plot point, and present one lowest rated option
       that points to every other child plot point.
7:     Collect the player's ratings and include them into $r_p$ and $r_o$.
8:     Collect the player's option selection. Present the corresponding child
       plot point to the player.
9: end while
```

Fig. 11. The complete drama management algorithm.

patterns for all the options in the branching story graph. As in the case of prefix preference learning in Section IV-B, we ask players to rate all the options presented after each plot point in the training phase, and construct an option-rating matrix. The option-rating matrix will be similar to the sparse prefix-rating matrix in Figure 3. Each row of the option-rating matrix contains one player's preference ratings for all the options while each column contains ratings for one option from all the players. The option-rating matrix also contains a lot of missing ratings. To train the player model on the option-rating matrix we use NMF and pPCA (as in Section IV-C) as well as K-Nearest Neighbor and K-mean algorithms. The learned player model retains the extracted rating patterns for players of different option preference types and will be used to predict future players' preference ratings over the options.

B. Drama Manager Algorithm

Our drama manager requires a player model for story/prefix preferences as described in Section IV and the player model for option preferences as described above. The drama manager algorithm is shown in Figure 11.

The interactive narrative experience begins with a root plot point. For each plot point presented to the player, the drama manager uses the prefix rating prediction to pick the child plot point that will lead to the highest rated descendant leaf node in the prefix tree. Given a player model for option preferences, the manipulation of options presented to the player is straightforward. The drama manager presents to the player the two options that lead to the desired child plot point that are predicted to be highest rated by the player. For each non-desired child plot point, the drama manager also presents the option predicted to be rated lowest by the player. This option presentation scheme maximizes the likelihood that the player will pick the option leading to the desired child plot point, given the player model. It does not guarantee that options leading to the desired plot point will be higher rated than options leading to other plot points, nor that the player will pick the option leading to the desired plot point.

Note that it is not strictly necessary to collect option and prefix ratings as in step 7 of Figure 11; we do it in our system for the purpose of collecting as much data as possible to build more accurate player models. With every new rating, the DM will get better understanding of the current player's preference over the story prefixes and options.

VII. Preliminary Evaluation of Drama Manager

We have performed a set of preliminary human study to evaluate our algorithm's ability to guide players to a selected plot point. Instead of directly presenting the recommended plot points to the players as in Section V, we will give the players the appearance of full agency and let them choose the options. We test the hypothesis that the drama manager can significantly affect the branches chosen by interactive players.

A. Story Library

We transcribed the Choose-Your-Own-Adventure book, *The Abominable Snowman*, into our story library representation. As before, we modified the graph such that all possible narrative trajectories contain exactly six plot points. There are 28 leaf nodes and 19 branching points in the branching story graph. Figure 12 shows the branching story graph we used in our study.

For experimentation, we selected one branching point (plot point 2) from the graph. The text for plot point 0 and plot point 2 is shown in Figure 13, which is summarized as follows: the player must decide whether to continue to search for the Yeti or go elsewhere to photograph tigers. Plot point 2 has two children plot points. We authored a total of seven options for this plot point:

- *Carlos could be in danger. You decide to go ahead with the expedition for Yeti.*
- *You continue the expedition for Yeti since you could be the first person in the world to get Yeti photos.*
- *You have been preparing for the expedition for such a long time that you decide to continue to search for Yeti.*
- *It is wise to play it safe. You decide to postpone the expedition.*
- *You go for tigers since you can make a good fortune by selling photos of tigers in the Terai region.*
- *You go for tigers since many friends of yours strongly recommended photography in Terai region.*
- *Mr. Runal is an expedition expert. You follow his advice to go on the expedition for the tigers.*

The first three options lead down one branch in which the player continues to search for the Yeti, and the last four lead down another branch in which the player searches for Tigers. Each option emphasizes a different motivation (friendship, fame, consistency, safety, money, following others, and safety) that a player might value. We did this as our first step to evaluate the algorithm. Other theories from the area of psychology can also be used to author the options [29]–[32]. Our algorithm can be applied directly to handle more options created on other theories. In fact, more options will increase the likelihood that some options leading to the desired plot point will be preferred by the player.

B. Experiment 1: Training the Option Preference Model

The human study is divided into a training phase and a testing phase. In the training phase, we recruited 39 players

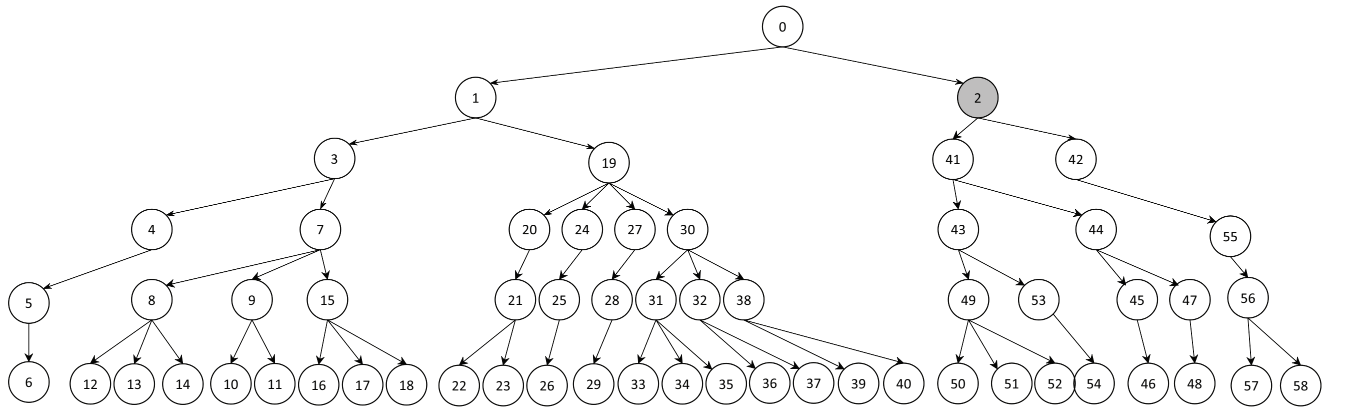


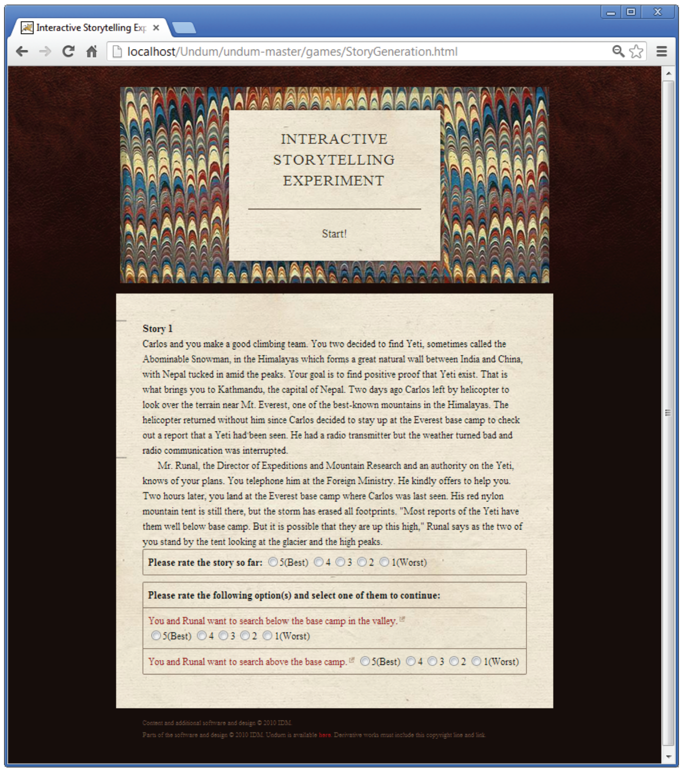
Fig. 12. The branching story graph for the choose-your-own-adventure book: *The Abominable Snowman*. The highlighted node (plot point 2) is the one we experimented on.

Carlos and you make a good climbing team. You two decided to find Yeti, sometimes called the Abominable Snowman... Two days ago Carlos left by helicopter to look over the terrain near Mt. Everest, one of the best-known mountains in the Himalayas. The helicopter returned without him since Carlos decided to stay up at the Everest base camp to check out a report that a Yeti had been seen. He had a radio transmitter but the weather turned bad and radio communication was interrupted... You have an appointment to speak with Mr. Runal, the Director of Expeditions and Mountain Research for the Nepalese government and an authority on the Yeti. You want to ask for his opinion at first... "Recently, a large expedition set out without telling us that they were going after the Yeti," says Runal. "They used guns and traps, and tried to kill one of them. The Yeti are angry. I must advise against you into the Yeti territory—I could arrange a trip for you into the Terai region. You could photograph and study the tigers. Later, perhaps, you could conduct the expedition you are leading."

Fig. 13. Plot point 0 and 2 from *The Abominable Snowman*

from Amazon Mechanical Turk. Each player read 5 full-length stories, presented plot-point by plot-point. After each plot point, the players rated the story-so-far and all the options on a scale of 1 to 5 before they could select one of the options to continue. Figure 14 shows a screenshot of our online interactive storytelling testbed. At the experimental branch with seven options but two children, four options (two for each child plot point) were randomly picked and presented to the player. Participants were asked to explore the graph as much as possible. If they encountered a plot point they had seen previously, their previous ratings for story-so-far and options were automatically filled out from their previous response. No participants chose to change their ratings upon encountering a plot point a second time.

To analyze our system, we randomly selected 80% of the players and collected their ratings into an option-rating matrix. We compute a player model by training CF algorithms on the option-rating matrix. The player model is used to predict players' preference over the options for the remaining 20% of subjects. The option preference is then used to predict players' selections at the experimental branching point which we selected in previous section. We repeated this process 50 times. Table V shows the results of the average prediction accuracy of option selection at the experimental branching plot point. We implemented four algorithms to predict players' option preference: pPCA and NMF (with dimension 4) as


Interactive Storytelling Ex
localhost/Undum/undum-master/games/StoryGeneration.html
INTERACTIVE STORYTELLING EXPERIMENT
Start!

Story 1
Carlos and you make a good climbing team. You two decided to find Yeti, sometimes called the Abominable Snowman, in the Himalayas which forms a great natural wall between India and China, with Nepal tucked in amid the peaks. Your goal is to find positive proof that Yeti exist. That is what brings you to Kathmandu, the capital of Nepal. Two days ago Carlos left by helicopter to look over the terrain near Mt. Everest, one of the best-known mountains in the Himalayas. The helicopter returned without him since Carlos decided to stay up at the Everest base camp to check out a report that a Yeti had been seen. He had a radio transmitter but the weather turned bad and radio communication was interrupted.

Mr. Runal, the Director of Expeditions and Mountain Research and an authority on the Yeti, knows of your plans. You telephone him at the Foreign Ministry. He kindly offers to help you. Two hours later, you land at the Everest base camp where Carlos was last seen. His red nylon mountain tent is still there, but the storm has erased all footprints. "Most reports of the Yeti have them well below base camp. But it is possible that they are up this high," Runal says as the two of you stand by the tent looking at the glacier and the high peaks.

Please rate the story so far: ○ 5(Best) ○ 4 ○ 3 ○ 2 ○ 1(Worst)

Please rate the following option(s) and select one of them to continue:

You and Runal want to search below the base camp in the valley.
○ 5(Best) ○ 4 ○ 3 ○ 2 ○ 1(Worst)

You and Runal want to search above the base camp. ○ 5(Best) ○ 4 ○ 3 ○ 2 ○ 1(Worst)

Content and additional software and design © 2010 IDM.
Parts of the software and design © 2010 IDM. Undum is available here. Derivative works must include this copyright line and link.

Fig. 14. A screenshot of the interactive storytelling system. A plot point is a paragraph of the story in the figure. Compared to Figure 7, multiple options are presented after every plot point. The story will unfold depending on players' selections of options.

described in Section IV-C, K-Nearest Neighbor algorithm (with K equaling to 2), K-mean algorithm (with k equaling to 3). As we can see from the table, pPCA achieves the best prediction accuracy.

C. Experiment 2: Testing the Drama Manager

In the testing phase, we recruited additionally 27 player-s from Amazon Mechanical Turk. The participants played through five stories. For an individual participant's first four trials, he or she was forced to explore the left side of the Figure 12 (the subtree with plot point 1 as its root). From

TABLE V
THE PREDICTION ACCURACY OF OPTION SELECTION AT THE EXPERIMENTAL BRANCHING PLOT POINT.

Algorithm	Accuracy
pPCA	0.8242
NMF	0.7626
KNN	0.7394
KMean	0.7367

TABLE VI
THE EFFECT OF PERSONALIZED DRAMA MANAGEMENT ON BRANCHES FOLLOWED.

Algorithm	% Yeti	% tigers	p-value
Without intervention	0.722	0.278	–
DM Intervention (target: Yeti)	0.95	0.05	<0.0025
DM Intervention (target: tiger)	0.33	0.66	<0.005

that point on, the protocol was the same as for the training phase: participants choose any option they desired. On the fifth trial, the participant is forced to go through the experimental point (plot point 2) and could choose any option from plot point 2 and on. On the fifth trial at the experimental plot point the drama manager would present three of the seven options: two of them pointing to the intended child plot point and one pointing to the other child plot point based on the DM prediction. Table VI shows the percentage of participants who followed each branch succeeding the experimental plot point with and without drama manager intervention. The first row of the table shows how often participants chose one branch versus the other during the training phase. The subsequent rows show the percentage choosing each branch when the DM was configured to guide players toward the Yeti or tigers, respectively.

Without intervention, players overwhelmingly prefer to continue to search for the Yeti (72.2%). When the drama manager is configured to direct players toward the Yeti branch, the probability that players will choose an option leading to that branch increases to 95%. When the drama manager is configured to direct players toward the other branch, the probability that players do so increases from 27.8% to 66%. All changes due to drama manager intervention are statistically significant.

D. Discussion

Player agency is a critical aspect of interactive narrative. The studies demonstrating the effectiveness of PBCF necessarily removed player agency in order to focus on the accuracy of story recommendations without the uncontrolled variable of player choices. Our drama manager maintains the appearance of player agency by carefully selecting options to present to the player based on predictions of his or her preferences over all options.

While our preliminary study did not address the question of whether the drama manager could guide the player down branches predicted by PBCF to be preferred by the player, we were able to show that the drama manager can significantly influence the trajectories of players, regardless of how the target branch is chosen. Future work will be to assess PBCF and the drama manager working fully together.

We did not ask players whether the drama manager reduced player perception of agency. However, since players are presented with options for all possible branches—no branches are outright denied to players—we believe that the appearance of player agency will be upheld.

VIII. Conclusions

Personalized drama management aims to personalize individual players' experiences in virtual worlds and games while having those experiences conform to a human designer's intent. We present two novel contributions to this end. First, the prefix-based collaborative filtering algorithm solves the *sequential recommendation problem* to predict the trajectories through a branching story graph of plot points that will be most preferred by an individual player. Our experiments demonstrate that we can learn a player model and use it to enhance players' experiences above that of an unguided walk through a branching story graph.

Second, we present a drama manager that can guide players along different trajectories through a branching story graph by predicting the choices at each branching point players will make and then manipulating which options will be presented to the player. Our preliminary study suggests that our drama manager can significantly affect the paths through the story that players take without reducing player agency.

Branching story graphs are simple, yet effective means for human designers to explicitly assert their authorial intent for what interactive narrative experiences are acceptable. We note that many drama management approaches that do not allow for personalization use different narrative content representations, but that these systems generate structures that are equivalent to branching story graphs. In the future we envision our drama manager operating in conjunction with more sophisticated interactive narrative systems and story generators.

We believe that Drama Managers should take player preferences into consideration when managing the unfolding narrative in a virtual world or game, thus balancing the player's interests against that of the human designer. The PBCF algorithm is a step toward incorporating players' individual differences into the gaming experience for the purpose of delivering more enjoyable experiences on an individual level.

Acknowledgment

We gratefully acknowledge the support of the U.S. Defense Advanced Research Projects Agency (DARPA) for this research.

References

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

[2] M. Riedl and R. M. Young, "From linear story generation to branching story graphs," *IEEE Journal of Computer Graphics and Animation*, vol. 26, no. 3, pp. 23–31, 2006.

[3] J. Bates, "Virtual reality, art, and entertainment," *Presence: The Journal of Tele-operators and Virtual Environments*, vol. 1, no. 1, pp. 133–138, 1992.

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

[5] 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.

[6] 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.

[7] 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.

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

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

[10] B. Li and M. O. Riedl, “An offline planning approach to game plotline adaptation.” *Proceedings of the Sixth Artificial Intelligence and Interactive Digital Entertainment conference*, 2010.

[11] B. Medler, “Using recommendation systems to adapt gameplay,” *International Journal of Gaming and Computer Mediated Simulations*, vol. 1, no. 3, pp. 68–80, 2008.

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

[13] 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.

[14] 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.

[15] 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.

[16] R. Lopes and R. Bidarra, “Adaptivity challenges in games and simulations: A survey,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 2, pp. 85–99, 2011.

[17] 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.

[18] 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.

[19] M. S. El-Nasr, “Engagement, interaction, and drama creating an engaging interactive narrative using performance arts theories,” *Interactions Studies*, vol. 8, no. 2, pp. 209–240, 2007.

[20] B. Li, S. Lee-Urban, D. S. Appling, and M. O. Riedl, “Automatically learning to tell stories about social situations from the crowd,” *Proceedings of the LREC 2012 Workshop on Computational Models of Narrative*, 2012.

[21] M. S. El-Nasr, “Interactive narrative architecture based on filmmaking theory,” *International Journal on Intelligent Games and Simulation*, vol. 3, no. 1, 2004.

[22] R. S. Sutton, “Temporal credit assignment in reinforcement learning,” *Electronic Doctoral Dissertations for UMass Amherst, Paper AAI8410337*, 1984.

[23] K. Yu, S. Zhu, J. Lafferty, and Y. Gong, “Fast nonparametric matrix factorization for large-scale collaborative filtering,” *Proceedings of the 32nd SIGIR conference*, 2009.

[24] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society*, vol. B61, no. 3, pp. 611–622, 1999.

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

[26] 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.

[27] [Online]. Available: http://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

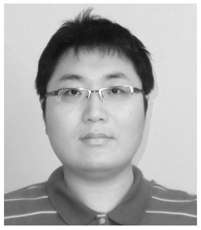
[28] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006. [Online]. Available: www.GaussianProcess.org/gpml

[29] R. B. Cialdini, *Influence: The Psychology of Persuasion (Collins Business Essentials)*. Harper Collins Publishers, 2006.

[30] R. Figueiredo and A. Paiva, “”I want to slay that Dragon!” - Influencing choice in interactive storytelling,” *Proceedings of the Third Joint Conference on Interactive Digital Storytelling*, 2010.

[31] ——, “Persu - An architecture to apply persuasion in interactive storytelling,” *Proceedings of the 8th International Conference on Advances in Computer Entertainment Technology*, 2011.

[32] D. L. Roberts, M. L. Furst, and C. L. Isbell, “Using influence and persuasion to shape player experiences,” *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games*, 2009.



Hong Yu received the B.S. degree and M.S. degree in computer science from Shanghai Jiao Tong University in China. He is currently working towards the Ph.D. degree in School of Computer Science at Georgia Institute of Technology. He is a member of the Entertainment Intelligence Lab in Georgia Tech. His research interests include player modeling for computer games, story generation, artificial intelligence and machine learning.

Mark Owen Riedl is an Assistant Professor at the Georgia Institute of Technology School of Interactive Computing. Dr. Riedl’s research resides at the intersection of artificial intelligence, storytelling, and virtual worlds, focusing on how intelligent systems can autonomously create engaging experiences for users in virtual worlds. His research has implications for entertainment computing, virtual educational environments, and virtual training. He has published over 80 scientific articles on artificial intelligence, story generation, interactive virtual worlds, and adaptive computer games. His research is supported by the NSF, DARPA, the U.S. Army, Google, and Disney.