A Survey and Analysis of Techniques for Player Behavior Prediction in Massively Multiplayer Online Role-Playing Games

**BRENT HARRISON[1], STEPHEN G. WARE[2], (Member, IEEE),
MATTHEW W. FENDT[3], AND DAVID L. ROBERTS[4]**

[1]School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA
[2]Department of Computer Science, University of New Orleans, New Orleans, LA 70148 USA
[3]Department of Computer Science, Baylor University, Waco, TX 76706 USA
[4]Department of Computer Science, North Carolina State University, Raleigh, NC 27695
CORRESPONDING AUTHOR: B. HARRISON (brent.harrison@cc.gatech.edu)

ABSTRACT   While there has been much research done on player modeling in single-player games, player modeling in massively multiplayer online role-playing games (MMORPGs) has remained relatively unstudied. In this paper, we survey and evaluate three classes of player modeling techniques: 1) manual tagging; 2) collaborative filtering; and 3) goal recognition. We discuss the strengths and weaknesses that each technique provides in the MMORPG environment using desiderata that outline the traits an algorithm should posses in an MMORPG. We hope that this discussion as well as the desiderata help future research done in this area. We also discuss how each of these classes of techniques could be applied to the MMORPG genre. In order to demonstrate the value of our analysis, we present a case study from our own work that uses a model-based collaborative filtering algorithm to predict achievements in World of Warcraft. We analyze our results in light of the particular challenges faced by MMORPGs and show how our desiderata can be used to evaluate our technique.

INDEX TERMS   Computational modeling, games, machine learning, data mining, performance evaluation.

I. INTRODUCTION

As massively multiplayer online role-playing games (MMORPGs) become more popular, game designers look for new ways to innovate the genre in order to draw players to their product. One way to facilitate this innovation is to incorporate *player models* into the MMORPG genre in some fashion. The term player modeling, as we use it in this paper, refers to a *predictive, computational model of player behavior*. The subject of player modeling in games has been well studied over the years; however, research on player modeling is typically just applied to single player games or small-scale multiplayer games. In these studies, researchers have used player models to adapt gameplay for specific player types, generate content that more players would find satisfactory, and even discover level design mistakes during game production. It is not necessarily clear, however, how these techniques can translate from the single player environment to the MMORPG environment.

In this paper, we present a desiderata that can be used to evaluate the effectiveness of player modeling techniques in a MMORPG environment. We also present a case study which shows how player modeling techniques can be used in MMORPG environments. It also shows how our desiderata can be applied to player modeling techniques to determine their practicality in MMORPGs.

Finally, we survey various player modeling techniques and use our desiderata to outline their strengths and weaknesses with respect to their performance in an MMORPG environment. We will also describe possible applications they may have in the MMORPG genre. Specifically, we focus on how these techniques can be used to improve player experiences through improving the design of the game. Discussion of other possible applications of player modeling in MMORPGs (such as for bot detection or cheat detection) is beyond the scope of this paper. The three classes of techniques that we will survey are

manual tagging, *collaborative filtering*, and *goal recognition*.

With this paper, we hope to show that player modeling in a MMORPG environment is fundamentally different than in a single-player environment and, thus, must account for a different set of requirements. We hope that this review along with our desiderata will be useful to future researchers interested in designing player modeling algorithms for MMORPG environments.

The remainder of this paper is organized as follows: In Section II discuss in greater detail the desiderata that we will use to evaluate various player modeling techniques in MMORPG environments. Section III outlines our case study, in which we use a collaborative filtering algorithm to predict player achievements in World of Warcraft. This section also gives a concrete example of how our desiderata might be used to determine a player modeling technique's effectiveness in a MMORPG environment. In Section IV we survey four real world examples that we will use to illustrate how player modeling techniques can be applied to MMORPGs. Finally, we review and evaluate manual tagging, collaborative filtering, and goal recognition in sections 5, 6, and 7 respectively.

II. DESIDERATA

MMORPG environments present several challenges that do not exist in single player games or small-scale multiplayer games. The sheer size of the environment and the number of players that are allowed to interact with the world simultaneously places strict requirements on the types of techniques that can be used to predict player behavior in games. By considering the many challenges that exist in an MMO environment, we came up with the following set of desiderata that we will use in the remainder of this paper to evaluate player behavior prediction techniques with respect to their applicability to MMORPGs:

- Scalability
- Ability to Handle New Data
- Authorial Burden
- Performance on Unsupervised Tasks
- Noise Tolerance
- Accuracy

Each of these desideratum correspond to a challenge inherent in developing algorithms for player behavior prediction or player modeling in an MMO environment. Therefore, these are qualities that are highly desirable in techniques for behavior prediction in MMORPGs. After we discuss each technique for behavior prediction, we will evaluate each technique based on how well each technique addresses these desiderata. The ratings that we give are as follows:

- ○: The technique does not address the desideratum or is not able to address it to the level that would be required in an MMO environment.
- ◑: The technique has the potential to meet the desideratum requirements in an MMORPG requirement, but

its ability to do so is largely implementation specific or otherwise depends on other factors.
- ●: This technique fully addresses the desideratum.

We will now discuss each of these desideratum in greater detail.

A. SCALABILITY

The most obvious challenge that an MMO environment offers is its size. At any time, hundreds of thousands of players could be interacting with a truly massive world at the same time. Each of these players are producing a large amount of data with each action they perform in game. In order for a technique to be successful in this environment, it must be able to quickly sift through a large amount of data and make predictions about future player actions in *real time*. In other words, techniques must be able to quickly make predictions and must be able to be quickly trained on large amounts of player data.

B. ABILITY TO HANDLE NEW DATA

Every time a player performs any action in an MMORPG, more data is generated. The ability to efficiently incorporate this data into a learning technique of some kind is important for making accurate predictions about player behavior. If it takes a long time to incorporate new data, then it is likely that by the time new models have been developed, the data that they were built off of will be old and its use will be limited.

A technique should also be able to adapt to the ever-changing environments that are common in modern MMORPGs. Content is constantly being added to these environments, and the last thing that a developer wants is to have to delay content release because the player modeling techniques in place cannot handle the release of this new content in an efficient manner.

C. AUTHORIAL BURDEN

Creating a model of player behavior can be a very difficult and very costly exercise. Creating an accurate model of player behavior can potentially be costly in multiple different ways. For example, it can be costly in terms of time it takes for a person to come up with possible player types by hand and then exhaustively list the possible actions that each type of player could take. On the other hand, it could be a significant financial cost if multiple writers are employed to ease the time investment required to perform such a task.

If one uses a computational model to describe player behavior, the creation of this model could incur a great deal of authorial burden if it requires a large amount of observational data to produce an accurate model. Since it can be difficult for some game designers to come by large amounts of player behavior data before a game is released, this can be seen as a different, yet equally important, type of authorial burden. Ideally, a player modeling technique should minimize the amount of effort that the game's author needs to put into creating the player models.

D. PERFORMANCE ON UNSUPERVISED TASKS

In machine learning and data mining, there exists the dichotomy between *supervised learning* and *unsupervised learning*. In a supervised learning problem, you are given training examples that are labeled with whatever behavior you want to predict. If you wanted to create a model of player behavior using a supervised learning method, for example, you would provide training examples that contained some in game behaviors that are then labeled with the player type associated with that example. In an unsupervised learning problem, training examples are not labeled, and it is up to the learner to determine how best to group examples into types.

Data in an MMORPG is inherently unsupervised as players do not often identify a player type before beginning play. Even if they did, there has been work done that calls into question the validity of self-report data [1]. Also, players will frequently not make the knowledge of what actions they are likely to complete next available during gameplay, which again lends credit to the idea that MMORPG data is unsupervised.

Since this is the case, techniques for player modeling in a MMO environment need to be able to handle unsupervised data. This can be done either by employing an algorithm that is able to handle this type of data (such as a clustering algorithm), or somehow intelligently converting the problem into a supervised learning problem (as is typically done when manual tagging is used).

E. NOISE TOLERANCE

One side effect of having possibly hundreds of thousands of players interacting with a virtual environment is that you are prone to receive *noisy* data. Data that is *noisy* is data that is difficult or impossible to interpret due to it being unstructured or being generated by a spurious source. Due to the size of a MMORPG's playerbase as well as the possible number of activities that a player can undertake, it is very likely that the data received from a player during gameplay will contain a large amount of noisy data. In this environment, it is important that algorithms are able to distinguish data that contains actual predictive trends (often referred to as a predictive *signal*) from that which is nothing but noise. If a technique is able to do this, we say that this technique is *noise resistant*.

F. ACCURACY

For a player model to be useful, it must be able to accurately predict player behavior. There are many definitions of what constitute player behavior, and could include anything from predicting player actions to predicting player personality types. In a MMORPG environment, if you are going to perform any of the tasks that we have mentioned earlier it is of the utmost importance that your predictions be accurate because it is oftentimes detrimental to the gameplay experience to make an incorrect prediction. This is because it could lead to tailoring content based on the assumption that this prediction is correct while it may, in reality, be wrong.

III. CASE STUDY

We have just discussed a desiderata for evaluating how player modeling techniques can be used in MMORPG environments. In this section, we will present the results of a case study in which we used a collaborative filtering algorithm to predict achievements that players are likely to complete in the popular massively multiplayer online role playing game (MMORPG) World of Warcraft (WoW). This is meant to be a high level overview of the case study in which we evaluate this work in terms of the desiderata described earlier. For a more detailed description of the work, please see [2]. We will discuss collaborative filtering in more detail in Section 6.

A. METHODOLOGY

In this study, we explored the use a collaborative filtering technique using clique-based graph clustering on a dataset consisting of 1289 achievements from World of Warcraft (WoW). *Achievements* are milestones that you can complete in games by performing certain, usually somewhat obscure, actions. Achievements can be obtained for doing many different things, so our reasoning was that this technique could be used to recommend content to players based on their achievement preferences. In other words, using this clustering algorithm, we wanted to be able to guide players towards achievements that we felt they would enjoy doing.

Our CF technique can be can be broken down into two high-level steps:

1) Build cliques of highly correlated achievements
2) Calculate the probability of completing achievements during gameplay given a player's achievement history

During the first step, a computational model of achievements is created by clustering achievements based on how likely they are to be completed together. This is done by first making a complete correlation graph of achievements. In this graph, nodes represent achievements and edges between nodes are weighted with the correlation value between those two achievements. Once this has been done we downselect edges so that only edges between highly correlated achievements remain. Next, we find all *maximal cliques* in this graph. A *clique* is a set of nodes that are all connected to each other. A *maximal clique* is the largest clique that is not the subset of another clique. Finally, we must downselect cliques to ensure that we have cliques that only contain achievements that players are likely to complete together. Since we use correlation, the resultant cliques contain achievements that players behaved similarly on. If many players completed all of the achievements together, they will be in a clique. If players did *not* complete the achievements together, they will also be in a clique. This downselection is performed in order to remove the latter type of cliques from the dataset.

In order to actually make predictions about which achievements a player is likely to complete, our method observes players as they move through the game. As they complete achievements, we calculate the probability $P(Q|A)$. This, informally, is the probability that a player will complete the

TABLE 1. Summary of results. Compares the performance of the clique-based models (my models) against models created by a random baseline. Average precision and recall values (± standard deviation) are reported across all runs of a 10-fold cross-validation. Also reported are results of t-tests between values for the clique models and the random models.

	$t = 0.6$		$t = 0.7$		$t = 0.8$	
	Precision	Recall	Precision	Recall	Precision	Recall
Clique	0.73 ± 0.14	0.46 ± 0.09	0.76 ± 0.12	0.42 ± 0.10	0.80 ± 0.14	0.36 ± 0.08
Random	0.20 ± 0.11	0.11 ± 0.03	0.21 ± 0.11	0.10 ± 0.03	0.20 ± 0.11	0.08 ± 0.02
Significance	$p < 0.05$	$p < 0.05$	$p < 0.05$	$p < 0.05$	$p < 0.05$	$p < 0.05$

remaining achievements in clique Q given that they have already completed the set of achievements A. This is calculated for every clique whenever an achievement is completed. When this probability surpasses a threshold value t for a given clique, the algorithm predicts that the player will complete the remaining achievements in that clique.

To test this, we built models on 7490 characters that were gathered from the WoW Armory, an online database of character information, using the WoWSpyder web crawler [3]. We then tested our method on 100 characters that we had not seen before. To quantify our algorithm's performance, we calculated *precision* and *recall*. Precision is the percentage of achievement predictions made for a player that were actually correct, and recall is the percentage of completed achievements for a player that were actually predicted by our algorithm. In order to observe what effect different threshold values would have on precision and recall values, we chose to run the experiments for varying values of t. Specifically, we chose $t = 0.6$, $t = 0.7$, and $t = 0.8$. In this study, we compared our algorithm against a random baseline model that made random achievement predictions.

B. RESULTS

The results of these experiments are summarized in Table 1. Contained within the table are mean precision and recall values for each algorithm across all values of t tested. Also contained are the standard deviations associated with each measure. In this study, we found that our models outperformed the random model algorithm across all thresholds. As can be seen in the table, our models produced higher precision and recall values compared to the random model. In each case, our models obtained much higher precision than recall. In order to further verify our results, we ran a one-sided t-test to verify that the difference we found between these precision/recall values was statistically significant. The test produced a p-value of $p < 0.05$ in all experiments, showing that there is, indeed, a statistically significant difference between the performance of the random model and the models generated by our CF technique.

C. DESIDERATA

In this section, we will discuss this case study in terms of the desiderata introduced earlier in this paper. This section is meant to provide a summary of the lessons that we learned while performing this study.

1) SCALABILITY
The two most computationally-intensive, and therefore time-intensive, steps in this algorithm are:

1) Creating the complete correlation graph
2) Finding all maximal cliques

It is not surprising that these two steps incur the steepest time cost. In order to create the complete correlation network, you must calculate all pairwise correlations for every achievement in your dataset. Our dataset contained 1289 achievements, which meant that we had to perform 830, 116 correlation calculations. In other words, this part of the algorithm does not scale very well with respect to the number of nodes in the correlation graph.

Finding all maximal cliques is an NP-complete problem, so this probably naturally scales poorly. To put things into perspective, these steps combined took about two days of computation to complete. Now, it is important to note that the most time-consuming steps of the algorithm are the steps that are performed *offline*. In this very limited example, it required 2 days of offline computation to complete. This is a cost that we were willing tolerate, and it is a cost that we believe is manageable for most MMORPG researchers. That being said, as the number of actions (achievements in this case) that a player can complete increases, the amount of time spent on this step will also increase.

2) ABILITY TO HANDLE NEW DATA
Adding new data to these models consists of periodically rebuilding the models. As we have previously discussed, building the models is a very time-consuming endeavor, so rebuilding the models should be done only when completely necessary. Typically, this means that the models should only be rebuilt when a large amount of new data has become available.

3) AUTHORIAL BURDEN
In general, the authorial burden of this technique is very low. Since these models are built from player observations, the author does not need to come up with player types or taggings out of nowhere. The fact that these models are built from observations, however, causes a different type of burden. This means that an initial round of data collection must be performed in order to generate the observations necessary to build the models. For researchers, this is typically not a problem since the data necessary will be acquired from

MMORPGs that have already been released. For those who wish to incorporate these techniques into their own games, it can be a little harder to generate the data necessary to create these models.

4) PERFORMANCE ON UNSUPERVISED TASKS

Since this is a clustering technique, it is designed to work on unlabeled data. In this case study, our only input was a set of characters and data concerning when they completed an achievement. By performing the clustering step, we group achievements based on correlation and, in a way, those become our player types. The second step involves determining if a new player belongs to any of the available player types. So, as you can see, this technique is well equipped to handle unsupervised tasks.

5) NOISE TOLERANCE

In this algorithm, the threshold parameter t plays a large role in making this algorithm noise tolerant. The t value increases the amount of confidence that our algorithm must have in its predictions before it makes them. For data that is noisy, it is unlikely that the required confidence level would be met in order to make a prediction. This phenomenon can actually be observed in our results. As the threshold value increases, the quality of our predictions, as measured by precision, increases. This performance increase comes at a cost, however. By increasing the threshold value, we require that the algorithm have more observations before a prediction is made. This means that the *number* of predictions made decreases.

6) ACCURACY

As you can see in our results, our precision values are significantly higher than our random baseline's. The reason that we are mainly concerned with precision is because precision shows the percentage of our predictions that were correct. Every time our algorithm makes a prediction, the understanding is that it would then act on this prediction in some way (by filtering content, for example). If this happens, you want to be sure that your predictions are correct since it is often *more* detrimental to the gameplay experience to encourage the player to do things that they do not want to do than to make no suggestions at all.

IV. APPLICATIONS TO MMORPGs

In order to help assess the applicability of each technique to MMORPGs, we have chosen to provide examples of possible applications found in MMORPGs that may benefit from player modeling. For each of these applications, we provide a concrete example that exists in a game and will use this example when evaluating each technique's applicability to the given problem. These applications are running examples that we use throughout the paper to show how player modeling techniques can be applied to preexisting MMORPGs. The applications that we consider are *interactive tutorials, targeted skill improvement, quest offering, and dynamic quests.*

Each of these applications rely on being able to predict player behavior in some way. For each class of techniques, we will also explore how that technique could be used for each of the above applications. In the following sections, we will review each of these topics in a little more detail.

A. INTERACTIVE TUTORIALS

MMORPGs typically offer players a handful of character classes to choose from. Class determines what abilities that character will develop as it advances and what role that character will take on when grouped with other players. The choice of which class to play has probably the single largest effect on what game content a player will experience, but this choice is almost always made before the player starts the game.

Several RPGs, most notably Bethesda's Elder Scrolls IV: Oblivion, offer the player an interactive tutorial at the beginning of the game to help them select an appropriate class. The player begins as an unknown prisoner who is given a chance to escape via a secret passage. Along that passage the player encounters low level monsters which can be overcome with weapons or spells. The player also learns to sneak, pick locks, and various other game mechanics. At the end of the tutorial, the game recommends a class to player based on which skills they used to overcome the challenges in the secret passage.

B. TARGETED SKILL IMPROVEMENT

In most modern MMORPGs, there is an emphasis placed joining together with other players to form a group in order to overcome difficult challenges. While in groups, players typically fall into one of three possible roles: tank, healer, or damage dealer. Tanks are typically durable characters that protect the other members of the party by having enemies focus all of their attention on him. Healers, as the name implies, keep the party healthy. The healer must pay special attention to healing the tank since they will be taking most of the damage. It is the damage dealers job to quickly dispatch enemies while the tank has them distracted.

To successfully perform each of these roles, players must use separate sets of skills. It has become increasingly popular to give players opportunities to practice these specific skills in a training-type environment. As a specific example of this, consider the Proving Grounds in World of Warcraft. Proving Grounds offer role-specific challenges that are meant to teach players how to tank, heal, or deal damage. In these challenges, players are paired with NPC companions with the task of overcoming a set of enemies with the player performing their desired role.

C. QUEST OFFERING

MMORPGs typically contain hundreds of possible quests that players can complete; however, it is often the case that only a subset of these are available to players at a given times. These can be filtered in a number of ways. Some examples of these include quests that can only be completed by specific player

classes or quests that can only be completed during certain holidays or other times of year.

The specific example of quest offering that we consider are daily quests in World of Warcraft. In WoW, there are certain quests that can be completed once per day. Typically, these quests are located in central locations (referred to as hubs) where players can pick up several daily quests at a time. In WoW, there is a set of quests that can possibly be offered at a given quest hub; however, only a subset of those quests are available on a given day. This means that it is unlikely that the same set of quests will be offered on two consecutive days.

D. DYNAMIC QUESTS

Normal quest structure in MMORPGs do not usually give the player much freedom in terms of choice. For example, if a player needs to collect some number of items, there is probably only one way to retrieve the specified items. It is becoming increasingly popular to incorporate choice into this traditional quest structure. These choices often give the player options in how to complete the quest. As a specific example of this, we consider the quest Settling Accounts from Star Wars: The Old Republic. In this quest, the player is asked to kill an accountant for a crime lord. Upon confronting the accountant, however, the player is given the choice to kill him and complete the quest, or spare him and use his expertise to steal money from a rival crime lord. Depending on the choice the player makes during this quest, they are offered presented with different content.

V. MANUAL TAGGING

The act of manual tagging can be described as the act of defining a typography of players and then determining how specific actions in game reflect each individual type in this typography. A player typography is a division of players based on some discerning criteria. Examples of this criteria include separating players by playstyle, motivations for play, and skill. In order to come up with a player typography, one typically consults a *domain expert* and then uses insights garnered from this domain expert to discern what possible player types exist in game. This *domain expert* could be someone who is intimately familiar with player behavior, such as a behavioral psychologist, or even someone who is simply familiar with the genre that a particular game exists in, such as a game designer or even the author of the player models. Once this has been done, then the author must determine how every action available in the game contributes or detracts from each of the derived player types.

Despite its mechanical simplicity, this technique has remained quite popular and examples can be found in many AAA game titles. In *Star Wars: The Old Republic*,[1] for example, Bioware uses a simple manual tagging scheme for filtering content. In this scheme, two player types exist, *dark side players* and *light side players*. While performing actions in the game, a player is given several decisions that dictate

[1] http://www.swtor.com/

which type he or she belongs to. These decisions are manually classified into *dark side* and *light side* actions by the developers. If the player chooses to complete a quest by performing dark side actions, for example, they will probably complete the quest by using brute force methods that could endanger innocent NPCs. On the other hand, if a player chooses to complete a quest by performing light side actions they will probably be presented with content that provides a subtler, or less violent at the very least, approach to complete the quest. In this scheme, Bioware drew on knowledge contained in the genre, the *Star Wars* universe in this case, to determine the possible player types and then manually tagged which actions were *dark side actions* and which actions were *light side actions*.

Most player typing techniques that take advantage of manual tagging follow this template. The main difference between techniques comes from where the expert knowledge is coming from. Sometimes, the expert tries to take a well known behavioral theory and apply it to games, whereas other times the expert may simply observe gameplay and interpret how this behavior translates into discrete player types.

A. MANUAL TAGGING EXAMPLES

One of the first attempts to classify players into distinct types was done by Richard Bartle [4]. In this work, Bartle relies on his own observations of players in a multi-user dungeon (MUD) to determine how best to partition them. He divides players into 4 groups based on their motivations for playing:

- Achievers: Players that place the most value on acquiring in-game rewards and making progress in the game
- Explorers: Players that place the most value on exploring the virtual world as well as exploring the capabilities of the game engine
- Socializers: Players that place the most value on interacting with other players
- Killers: Players that place the most value on interfering with the gameplay of others

Bartle also defines a set of possible actions that could be associated with each of these player types.

In 2005, Chris Bateman *et al.* [5] derived a set of player types based on the Myers-Briggs typology [6]. The Myers-Briggs typology is based on a set of four dichotomies: extroversion-introversion, sensing-intuition, thinking-feeling, and judging-perceiving. A player's personality is defined through their values for each of these dichotomies. As with Richard Bartle, Bateman *et al.* were able to divide players into 4 distinct player types:

- Conqueror: These players are driven to overcome all challenges the game presents them and have others recognize them for their achievement
- Manager: Thse players view games as a problem and seek to discover strategies and develop skills in order to solve it
- Wanderer: These players are looking for a fun experience that they can use to escape their daily life

- Participant: These players want to feel like they are a member of both the game world as well as the larger game community

Each of these types encompasses 4 of the types available in the Myers-Briggs typology.

Ryan Houlette [7] describes a technique for creating player models that consists of creating a tree structure where the leaves represent all of the available actions that a player can take. Parents of these actions correspond to the different types of gameplay that contain these actions. For example, a player model that describes stealthy gameplay would consist of a tree which the leaves of the ''stealthy gameplay'' node would be actions such as *uses smoke grenades* and *avoids guards*. So, in order to use this technique, one would first have to create a set of trees to describe how each action contributes to each possible playstyle in the game.

In the PaSSAGE system [8], Thue *et al.* uses player models that were generated by examining Robin's guide for pen-and-paper role playing games [9]. In this case, Thue *et al.* derived a set of 5 player types from this text:

- Fighters: These players prefer combat and to take aggressive actions in game
- Power-Gamers: These players prefer to gain special items and riches
- Tacticians: These players prefer to think creatively
- Storytellers: These players prefer complex plots
- Method Actors: These players prefer to take dramatic actions

Thue *et al.* tagged choices that the player would make in the game with the player type that would feasibly most enjoy that option. They would keep track of which types of actions the player had taken, and would use this to determine which choices to offer the player.

B. EVALUATION

- **Scalability**: ●, All of the work involved in using this technique takes place during production and not actually at run time. While people are playing the game, determining how certain actions contribute to a player model is a simple lookup.
- **Ability to Incorporate New Data**: ○, If new content is generated for the MMORPG, then it must go through the same tagging process that occurred during initial game production. If a substantial amount of content is added, then this task quickly becomes too cumbersome to finish in a reasonable amount of time.
- **Authorial Burden**: ○, Time must be invested to both come up with the player types in the game and to actually tag every action with these player types, with most of the time being spent during the actual tagging process. Given the amount of content that exists in most modern MMORPGs, the amount of time it would spend to tag all of it is quite infeasible given the value that you would get out of the models.

- **Performance on Unsupervised Tasks**: ○, Manual tagging deals with unsupervised data by turning it into a supervised problem. The process of tagging every action with an associated player type is equivalent to adding a class label to unsupervised data.
- **Noise Tolerance**: ○, Manual tagging techniques consider all data concerning player actions to be relevant which makes it highly susceptible to noisy data.
- **Accuracy**: ◑, The ability for manual tagging techniques to accurately describe player behavior depends solely on the quality of the expert knowledge that was used to tag the data. If this expert knowledge is flawed in some way, then any predictions made using these tags will also be flawed. If the knowledge is accurate, however, then it is likely that any predictions made using the tags will be accurate.

C. APPLICATIONS TO MMORPGs

Manual tagging offers a very simple way to implement player modeling into MMORPGs. In the following sections, we will show how player modeling can be implemented using manual tagging in four examples taken from current AAA titles.

1) INTERACTIVE TUTORIAL

Manual tagging techniques are often used to create tutorials such as the one found in Oblivion. This is done by tagging each activity that a player can perform in this tutorial with each class and then observing the player as they complete the tutorial. By observing how the player interacts with the tutorial, recommendations can be made by simply recommending the class associated with the majority of the actions that the player took.

This is a popular technique because it is easy to implement, but it assumes that the author is correct about the initial tagging. The technique does is not as effective if the assignment of actions to classes is flawed in some way.

2) TARGETED SKILL IMPROVEMENT

Applying manual tagging to the Proving Grounds in WoW involves having an expert with prior knowledge or an author (such as the game developer) to manually identify what actions or attributes each role (tank, healer, or damage dealer) should exhibit. These can then be turned into the specific challenges that players will encounter when they take part in the Proving Grounds. This allows the developer to control the difficulty of the challenges and how it should progress. It is interesting to note that this is very similar to how it is currently implemented in WoW.

Using manual tagging to implement the Proving Grounds is relatively straight forward, and only requires that an author determine what each attribute of healing, damage dealing, or tanking should be focused on with each challenge as well as how the difficulty of each challenge should progress. What this technique lacks, however, is the ability to personalize the challenges that each player faces so that they improve upon skills that need improving. In terms of our desiderata,

this hurts the accuracy of our model since it is not able to distinguish the needs of individual players.

3) QUEST OFFERING

In order to use manual tagging to implement daily quests in WoW, you could use a system that defines a player model in terms similar to those we have seen above. In other words, each quest would be tagged with a specific player type, however the author wants to define them. Depending on the quests that the player completes, their own player model will change. When the player goes to pick up a set of daily quests, the game would only offer the quests that best fit each individual player's type.

This requires much more effort on the part of the author because in this case every quest in the game has to be tagged in order to determine how it should effect each player model. In most MMORPGs, this is a nontrivial amount of authoring which could lead to manual tagging becoming intractable for this problem, meaning that this model does not scale with the size of the game. If these resources are available, however, it is a relatively simple way to offer player specific content.

4) DYNAMIC QUESTS

The quest in SW:TOR, *Settling Accounts*, can be implemented similarly to how one would handle daily quests in WoW. In this case, previous choices or quests would need to be tagged and the quests that players completed in the past and the choices they made would contribute to the model of their behavior. As with daily quests in WoW, these would all have to be authored to determine just *how* they would effect the the player model. With this, it is possible to eliminate the explicit choice that the player makes and simply customize the quest to suit that player. So in the SW:TOR quest *Settling Accounts*, it is determined that the player is most likely to ally with the man they are supposed to kill, the quest can make this the ultimate goal without actually giving the player the choice.

In this instance, manually tagging every quest as well as every choice could be a large amount of work depending on the size of the MMORPG. If resources are available that make this a feasible task, then it is relatively straight forward to implement.

VI. COLLABORATIVE FILTERING

Collaborative filtering (CF) is the technique of using preferences of known users or populations to make predictions of preferences for an unknown audience. One well known application of CF is in commercial services with heavy traffic such as eBay, Amazon.com, and Netflix [10]. For example, Netflix will make recommendations on movies to watch based on a user's viewing history. CF has also been extended to making recommendations in games to make predictions about a player's desired narrative experience [11] and to make out-of-game recommendations in MMORPGs [12], [13].

The collaborative filtering umbrella breaks down into two specific approaches: memory-based CF techniques and

model-based CF techniques [10]. Memory-based CF stores all recorded examples in memory and then will query these examples directly in order to determine preferences. Model-based CF uses recorded data as input to a machine learning algorithm in order to make a computational model of user preferences. Regardless of the approach, all major CF techniques only have access to the user's action history when making predictions. In other words, CF techniques use only the user-item data and do not use features about the users (such as their age or gender) to make predictions [14]. In the following sections, we will review some of the ways to apply CF techniques to MMORPGs and then provide a more in-depth discussion of both memory-based and model-based CF techniques.

A. MEMORY-BASED COLLABORATIVE FILTERING

Neighborhood-based CF is a common memory-based CF algorithm where the weight or similarity of two users are computed and then a prediction is made using either a simple weighted average or a weighted average over all users compared to the target user [15]. The advantage of memory-based CF is its ease of implementation and performance on dense data sets, while its disadvantages include performance issues on large and sparse data sets, dependence on user ratings, and difficulty making recommendations for users that have not provided many observations for the system to use [10].

Due to the issue that memory-based CF techniques have with scaling to large datasets, these methods have not seen much use in the games community. That being said, there are a few notable counterexamples. Kyong Jin Shim *et al.* used an algorithm called PECOTA [16] in order to predict *performance* in Everquest II.[2] The PECOTA algorithm is typically used to predict the amount of home runs that a baseball player will hit in the current year. It works by looking at the player-in-question's past performance and compares it with the past performances of every player in a corpus. It then finds nearest neighbors and uses their future performances to generate a prediction. This is the very definition of memory-based CF except that it is used to predict home runs instead of preferences or ratings. In Everquest II, Shim *et al.* define *performance* as the time it takes to advance to the next level. This example is notable in that it used memory-based CF techniques on a large scale, MMORPG dataset; however, it is important to note that this study was performed offline since it is quite likely that it would have taken too long to be performed in a real-time setting.

Sharma *et al.* [17] used memory-based CF in order to predict player preferences in an interactive narrative environment. This technique used a nearest-neighbor approach that would examine how a player advanced the story in an interactive narrative, and then determine their enjoyment of the narrative based on ratings that other players with similar story paths and ratings gave their experience.

---

[2]http://www.everquest2.com/

Hingston *et al.* [18] present generative techniques for mobile games. They created InfiniteWords, where players are presented with images that they need to identify. The puzzles are generated with memory-based CF.

1) EVALUATION

- **Scalability**: ◑, In order to make predictions, memory-based CF methods must first search all observed data in order to find similar users. In a MMORPG, the size of this dataset will quickly surpass the amount of data that can be efficiently searched. Data structures such as K-D trees [19] have been used to speed up this retrieval step, but the size of data can still be an issue if it is especially large.

- **Ability to Handle New Data**: ●, New data is able to be instantly incorporated since it simply has to be added to the corpus of observations that is used to make predictions.

- **Authorial Burden**: ●, Typically, the algorithm used to make these predictions only needs to be implemented once. This is usually a very simple process and is not very time consuming, meaning that it does not add much work that the designers have to do to implement it.

- **Performance on Unsupervised Tasks**: ●, The collaborative filtering problem is inherently unsupervised since it typically operates on traces of actions/ratings made by many different users. Since this data does not contain a class label and is, therefore, unsupervised, all techniques used to solve this problem must be equipped to handle unsupervised data.

- **Noise Tolerance**: ◑, Techniques such as these are typically susceptible to noise; however, it is possible to modify the canonical CF algorithms in order to make them more noise resistant. One common approach to reduce the effect that noise has on predictions is to use an ensemble of many CF predictors instead of a single predictor [20], [21].

- **Accuracy**: ●, since collaborative filtering techniques take data generated from actual users into account when making predictions, it is likely that the predictions made will be accurate assuming low noise.

B. MODEL-BASED COLLABORATIVE FILTERING

Model-based approaches address the problem that memory-based CF methods have with scaling by constructing a computational model of training data in order to make predictions. Most of the time, using the model to make predictions is much faster than searching through an entire corpus of training examples which makes most model-based CF techniques scale better than memory-based ones. This can be done with Bayes Nets [22], [23], clustering models [24] or others [25], [26]. Model-bassed CFs tend to perform better than memory-based CFs in large data sets [27], [28]. While model-based techniques do scale better than memory-based ones, there is an added cost up front because the models need to be trained on observation data before they can be used.

This cost, however, is typically only incurred once and can be done off-line.

Zook *et al.* use a tensor factorization technique to predict a player's mastery of a skill in both military training scenarios [29] and in a game that emulated the combat system used in a turn-based role-playing game [30]. This technique uses a player's past performance at various skills and then predicts what their future performance will be. In this work, this knowledge was then used to generate missions that would effectively *teach* the user how to use a certain skill, making this type of technique very useful for an adaptive help system.

Yu and Riedl [11], [31] apply prefix-based CF to a Drama Manager which makes plot decisions in narrative games. The Drama Manager makes decisions about which plot points to include in the story and their ordering. The CF is trained by player feedback on story event ordering.

In the domain of MMORPGs, Li and Shi [12] use CF to recommend items in item stores and also models the satisfaction that is associated with said item purchase. The authors use an analytic hierarchy process combined with an improved ant colony optimization technique in order to quickly converge upon possible recommendations to make.

ThaiSon and Siemon [13] use a CF technique which clusters wiki pages for users to visit based on their play in MMORPGs. It is important to note, however, that this method was only used to cluster and make recommendations about websites related to a MMORPG and did not take into account the player's actions in-game. This type of system could feasibly be used to intelligently guide players to third-party sources of information about a game.

Min *et al.* [32] apply the model-based collaborative filtering methods of probabilistic principal component analysis (PPCA) and non-negative matrix factorization (NMF) to the domain of serious games. These techniques were used to predict student performance on a learning game called Crystal Island which teaches middle school microbiology. They found that PPCA provides the most accurate predictions on average but that NMF provides a balance between run-time efficiency and predictive power.

1) EVALUATION

- **Scalability**: ●, Model-based CF techniques scale much better than memory-based ones. Making predictions using a computational model is typically a fast process that is easily scalable to hundreds of thousands of users.

- **Ability to Handle New Data**: ◑, In order to incorporate new data into these models, they must be rebuilt. This can be a time-consuming process; however, one typically does not need to rebuild the model until a significant amount of new data has become available. This means that, while the computational models will need to be rebuilt, they do not need to be rebuilt every time a player performs any action.

- **Authorial Burden**: ●, While model construction might take some time to complete, the training algorithms do not require very much author intervention to run.

Also, model building is performed very few times. Overall, the use of these techniques requires very little effort on the part of the author in order to work properly.

- **Performance on Unsupervised Tasks**: ◐, As with memory-based CF techniques, model-based techniques are very well equipped to deal with unsupervised problems. This does mean, however, that the types of models you can construct will be limited to those that can handle unsupervised data, such as clustering techniques.
- **Noise Tolerance**: ◐, While model-based CF techniques are more noise resistant than memory-based techniques, they are still susceptible to noisy data. There are ways to minimize this issue, however, such as ensemble learning methods.
- **Accuracy**: ●, As with memory-based CF techniques, these methods use actual user data to make their predictions. This increases the likelihood that they make accurate predictions, especially when compared to methods like manual tagging, which do not make predictions based on player observations.

C. APPLICATIONS TO MMORPGs

In the following sections, we will show how CF techniques can be applied to the real-world examples that we have mentioned previously.

1) INTERACTIVE TUTORIAL

Collaborative filtering is well suited to for implementing the tutorial in Oblivion. At a high level, collaborative filtering would require training in which techniques would examine how other players interacted with the tutorial and what class they eventually chose. Using these observations, the game would make recommendations to the player based on how they played through the tutorial.

The main benefit of collaborative filtering is that it allows for behaviors that game designers may not have accounted for to be associated with different classes. For example, there may be more to being a thief than just being sneaky. Since collaborative filtering uses observed actions to make recommendations, it can account for these situations. In order to use these techniques, however, some amount of training is required. This means that a sometimes nontrivial amount of observations will be required before these techniques could be used.

2) TARGETED SKILL IMPROVEMENT

Collaborative filtering is also well suited for use in implementing the Proving Grounds in WoW. In this case, using a technique such as those employed by Zook *et al.* [29], [30] can be used to *train* the player how to perform as a tank, damage-dealer, or healer. The benefit of these types of techniques is that they can be used to customize mission sequences that are meant to maximize player performance gains. As with all collaborative filtering techniques, some amount of training is required which could be prohibitively costly in terms of time or other resources.

3) QUEST OFFERING

Modifying the WoW daily quest structure to incorporate collaborative filtering would involve examining each quest that a player completed and then offering quests that other players with similar quest histories have completed. Here, training would be prohibitively expensive in most cases. To perform this, you would need to have sufficient observations to explore the space of possible quest histories as well as the space of daily quests accepted. Both of these will require exponentially more observations as the number of quests grows. It is possible, however, to work around this in this environment. Instead of having a formal training phase, for example, there could be times were daily quests are offered randomly (for training) and times when daily quests are offered in accordance to the collaborative filtering model.

4) DYNAMIC QUESTS

Collaborative filtering can be used to determine the choice that the player will make in the quest *Settling Accounts*. By doing this, the player does not need to be offered a specific choice, which could make the quest feel more organic and not require a break in immersion to make an explicit choice.

Another possible application of collaborative filtering involves intelligently selecting choices in order to bring about the desired quest outcome. In [31], Yu *et al.* use collaborative filtering in a choose your own adventure story to intelligently present choices to the player to bring about a desired outcome. Collaborative filtering can be used to determine which choices a player is likely to make, which means that choices can be intelligently presented to the player in order to increase the probability that the player chooses that choice.

VII. GOAL RECOGNITION

Goal recognition is the task of abductively reasoning about the user's intentions based on their observed actions [33], [34]. It assumes that the user is engaged in goal-directed behavior—that is, the user is trying to place the world into some specific state. The task of goal recognition is to predict what state the user is trying to put the world into based on the actions they have taken so far and knowledge about the domain.

Goal recognition is closely related to the problems of action recognition and plan recognition. Action recognition [35] (also called activity recognition) is the low-level task of deciding what action a user is taking based on sensory information such as computer vision. Because a game's state is fully-observable for the developer and because game interfaces are usually semantically explicit, activity recognition is usually not needed in MMORPGs. In other words, we know what the user is doing but not why [36]. Plan recognition [33] is the more general and more difficult problem of predicting not only the user's final goal but also the exact plan or sequence of actions they will use to achieve it.

Goal recognition techniques can be broadly divided into two types: those based on planning systems and those based

on probabilistic models. While they both accomplish the same task, they have different technological limitations.

A. PLANNING-BASED MODELS

Early goal recognition systems (generally in the 70's and 80's such as [33], [37], and [38], but also as recently as 2010 [39]) generally used symbolic logical reasoning to deduce the user's goals. They were similar in many ways to planning systems, which construct chains of actions to explain how an agent can accomplish a goal. Planning-based systems require the designer to provide a detailed model of the domain and annotate which actions can lead to which goals, as well as the causal and temporal constraints that exist between chains of actions. As new observations of user actions are made, these systems narrow down the list of possible goals that the user might be pursuing that are consistent with the actions taken so far [34]. The more ''useable'' an action is when planning toward some goal, the more likely that action was taken in service of that goal.

Planning-based goal recognition systems are most suitable for low-level narrative mediation. Mediation is the process of rewriting a story when the player takes actions that make the current story impossible to carry out. Both reactive and proactive narrative mediation have been studied. Reactive mediation is the process of attempting to repair a story that has been broken by the player's actions [40]. Proactive mediation is the process of attempting to anticipate which player activities will break the story and find ways to prevent or support them in advance [41]. Both rely on a model of goal recognition to prevent the user from breaking the current story or to incorporate the user's desired actions into the story.

While narrative mediation may be the gold standard for quests in a persistent open-world MMORPG, planning-based goal recognition and narrative mediation are simply too computationally expensive to be done in real-time, even for a small number of users. Writing a planning domain to include all the constraints on all the possible actions that a player can take is too great of an authorial burden. Also, logical deductive systems like these are not very tolerant of noisy data. In short, planning-based goal recognition systems are probably not practical for use in MMORPGs any time in the near future due to their inability to scale to large scenarios.

1) EVALUATION

- **Scalability:** ○, While low-level narrative mediation may be the eventual goal for quests in a persistent open-world MMORPGs, most planning-based goal recognition and narrative mediation techniques are simply too computationally expensive to be done in real time, even for a small number of users. These issues of speed can be mitigated somewhat by using faster hierarchical planners [33].
- **Ability to Handle New Data:** ◐, Planning-based techniques are often domain-independent and so do not change significantly when their domain models are modified. However, adding new elements to a domain

model (*e.g.* new game mechanics or new content) often necessitates changes to existing elements.
- **Authorial Burden:** ○, The task of modeling all the actions in all the quests of a MMORPG to the level of detail required by a planning system would be a massive effort, and is thus probably impractical.
- **Performance on Unsupervised Tasks:** ●, The author of the domain model must annotate which states are valid goals. While this might be considered a supervised task, it is a trivial amount of extra work given the existing authorial burden. The main strength of planning-based goal recognition systems is that they do not require a training corpus. If a domain model can be produced along with the game, it can be deployed as soon as the game is released without the need for any preliminary data collection.
- **Noise Tolerance:** ○, Techniques based on deductive logic do not handle noise well.
- **Accuracy**: ◐, Many early planning-based systems were described as theories along with examples of how they would work. Most were not tested using a corpus of real-world problems, so it is difficult to gauge their accuracy. Due to their low tolerance for noise, the accuracy of planning-based techniques in MMORPGs is likely to be lower than desired.

B. PROBABILISTIC MODELS

When players can pursue multiple goals in a non-linear fashion, and when they may make mistakes along the way, goal recognition is a noisy and uncertain process. For this reason, most modern techniques are based on probabilistic methods. Charniak and Goldman were some of the first to use Bayesian Networks [42] for goal recognition, while Bui [43] used a variation on Hidden Markov Models to accomplish goal recognition in real time.

While these methods scale better, tolerate noise, and are potentially less onerous to the game designer, they sacrifice a level of narrative granularity. Planning-based approaches reason at the level of atomic actions and thus can mediate even the smallest part of a story. Probabilistic models require the narrative content to be broken down into individual pre-scripted chunks (*e.g.* scenes or chapters) which cannot be further customized and are difficult to parametrize. Another weakness of these methods is that they may have difficulty modeling the uncommon paths to a goal, and in doing so may create a feedback loop by guiding more players down the most-traveled paths at the expense of the least-traveled (but still valid) paths. However, most MMORPG designers will find these restrictions reasonable given the many benefits of probabilistic approaches.

The transition from plan-based models to probabilistic models happened gradually as deficiencies in early systems were addressed. One of the first advances in modern goal recognition was to replace the onerously hand-written planning domain with a corpus of plans and their associated goals. Statistical and learning models are able to infer the

temporal and causal constraints on low-level actions from these corpora when they are not explicitly provided by the author [40]. Blaylock and Allen [44] used such a corpus to tune Bayes' rule to use bi-grams of observed user actions to predict what goal the user was pursuing. Their approach runs quickly (linear in the number of possible goals) and can scale to a large game. Of particular interest is work by Albrecht, Zukerman, and Nicholson [45] who used Dynamic Bayesian Networks to predict the current goal, next action, and next location of a person playing a text-based online Multi-User Dungeon (the precursor to modern MMORPGs) based on a database of successful quests. Similar work by Mott, Lee, and Lester [46] used Bayesian Networks trained on a corpus of completed quests in an education game. Gold [47] used an Input-Output Hidden Markov Model to predict one of three high-level goals in an action/adventure game: explore, level up, or return to town. His IOHMM can be trained in real time, and it outperformed a hand-authored Finite State Machine based on expert knowledge. However, all these approaches rely on collecting a corpus of supervised data, which may still be too great of an authorial burden given the sheer number of quests in a typical MMORPG.

Orkin, Smith, Reckman, and Roy [48] describe one method to reduce this burden. They collected thousands of instances of human players acting out the roles of a waiter and a diner in their online *Restaurant Game*. They demonstrated that a small corpus of hand-annotated game logs can be used to annotate a larger corpus automatically.

Work by Ha *et al.* [49] also offer a promising solution to the problem of collecting supervised data. Their testbed environment is an educational game for middle school children in which the player must discover the cause of a spreading illness through various forms of investigation. Players can potentially adopt many different goals which are not explicitly assigned to them by the game. Ha *et al.* predicted a player's current goal using Markov Logic Networks, a combination of traditional logical deductive models and probabilistic graph-based models which have some strengths of both—chief among them the ability to produce much smaller and faster models for the same problems when compared to First Order Logic reasoning or Bayesian Networks alone. Ha *et al.* marked certain actions in their corpus of player logs as goals, but did not explicitly tag which actions were directed toward which goals. The system infers this from the corpus, significantly reducing the authorial burden. This system is also an excellent example of how goal recognition can be used in a game to support goals set by the player and to provide individually-tailored help when the player encounters difficulty.

The work by Ha *et al.* and most of the systems mentioned above make the assumption that the user is only pursuing one goal at a time. This is rarely the case in a MMORPG. Work by Geib and Goldman [50] used a hybrid of planning-based and probabilistic models to account for multiple interleaving goals, but their approach is at least NP-hard and thus would not scale for online use in a large game with many players.

Hu and Yang [51] used Conditional Random Fields to manage multiple interleaving goals, and while their current approach may not scale, they believe that it can be adapted to run in real-time.

Probabilistic models can also attend to the individual player. Lesh [52], [53] presents a recognizer-independent method for tailoring goal recognition to individual users based on their observed preferences. Gold [47] also demonstrated that, once a player is familiar with the game, that player's data can be used to train an Input-Output Hidden Markov Model which is more accurate for that specific player. Techniques like this can enable content which is not only adaptive based on the player's goals but also based on the player's personality and game history.

1) EVALUATION

- **Scalability:** ●, Probabilistic models require time to train, but once the model is built they can run quickly, even for a large domain. Many of these models can also be arbitrarily simplified (at the cost of accuracy) if they are too slow.
- **Ability to Handle New Data:** ◐, Most probabilistic models must be retained and rebuilt to incorporate new data. However, some models like Gold's [47] IOHMM can be updated in real time.
- **Authorial Burden:** ◐, While probabilistic approaches usually do not require a detailed domain model, they still require a corpus which may be difficult to obtain and annotate.
- **Performance on Unsupervised Tasks:** ◐, Even advanced probabilistic goal-recognition systems require the author to specify which states in a domain are goals. However, the relationships of actions to goals can be learned automatically.
- **Noise Tolerance:** ●, All probabilistic models can handle some degree of noise, and others can even be extended to handle complex interleaving goals.
- **Accuracy:** ●, With the shift to building models based on a corpus of real-world data came more robust evaluation metrics for those systems. Many probabilistic goal recognition systems perform well on the tasks set to them by their designers and should be adaptable to a MMORPG context. Most can be tuned to provide only high-confidence predictions if those are what is desired.

C. APPLICATIONS TO MMORPGs

In the following sections, we show how goal recognition techniques can be incorporated into our running examples.

1) INTERACTIVE TUTORIAL

The strength of goal-recognition systems lie in their ability to recognize why a player is performing low-level actions. They do not try to classify or stereotype players, and thus they are less useful for making high-level content decisions such as which class a player should choose.

However, if low-level goals can be associated with high-level content, these techniques may prove useful.

Consider the example of *Oblivion*'s interactive tutorial. If the game can recognize that the player is attempting to get past a hostile goblin with stealth rather than violence, the game might increase the likelihood of recommending Rogue over Warrior.

2) TARGETED SKILL IMPROVEMENT

Goal recognition is an excellent tool for providing targeted help to players who are trying to learn or improve [46], [52]. The Proving Grounds in *World of Warcarft* are currently designed as a set of benchmarks that players should strive toward in order to be effective tanks, healers, or damage dealers. They offer little specific guidance on how to pass those benchmarks. Training players to fulfill certain roles in a party could be done more intelligently if the game was able to critique the player's strategy and offer suggestions. Research in intelligent tutoring systems and education games (see [49], [54]) have demonstrated the effectiveness of providing player-specific guidance by inferring the user's specific difficulties using goal recognition.

3) QUEST OFFERING

Goal recognition techniques are not ideal for deciding which quests to offer a player for the same reason they are less applicable to interactive tutorials—they work with low-level actions and goals and are less useful for making high-level content decisions. However, as with interactive tutorials, if goals can be associated with a player's preferences, these techniques might have limited applicability. For example, Rogue players may prefer quests which have a stealthy solution.

4) DYNAMIC QUESTS

The typical MMORPG quest has a distinct moment when the player accepts the quest, followed by specific instructions for how to carry it out and a distinct moment of completion. Goal recognition is trivial in this paradigm because the player's goals are rigidly assigned, there is only one path to the goal, and actions taken in service of that goal rarely contribute to other goals. Reasoning about the player's intentions is fruitless. Improving the linear, vapid nature of these quests is one of the key design challenges faced by designers today [55], and goal recognition is an ideal tool for making quests more dynamic and adaptive.

For example, quests with a branching narrative structure can be guided by goal recognition. Rather than explicitly presenting the user with a list of choices in text, the game could infer what branch the player is taking based on their actions. Consider the "Settling Accounts" quest example. Rather than prompting the player to make an explicit choice between killing the accountant and teaming up with him, the game could infer that the player wants to kill the accountant when he or she attacks that character. This maintains the player's immersion in the narrative and increases the perception that the game is "playing along" and giving the player some degree of agency in the quest experience.

Goal recognition could also make the process of stringing quests or quest events together into whole stories more generalizable. "Settling Accounts" has a very rigid structure: a crime lord who has been cheated, an accountant, and a hideout for that accountant. These characters and locations are never reused despite the fact that they could potentially be applicable to a number of other stories. The consequences of the player's choices do not affect the game outside of the ending of that particular quest. The crooked accountant from "Settling Accounts" could be integrated other quests. Imagine a quest that involves forging financial records. Rather than assigning the player the specific goal of visiting one character in one location to forge the records, the game could reason at a higher level about the goal of forging those records. Rogue characters might be able to forge the records themselves. Others could visit any accountant in the area, including the one previously met in "Settling Accounts" (provided he was left alive). By reasoning more generally about the player's goals, a game can reuse assets while simultaneously creating quests with many possible solutions.

Goal recognition may also have practical benefits for typical MMORPGs in the short term. Tomai and Salazar [56] discuss the challenges of managing player quests in a shared persistent world. Their system changes where each player is sent and what monsters each player is asked to kill to avoid creating too much competition and thus reducing the frustration that players often experience when vying for shared resources. Even without changing the basic quest structure, an adaptive game can anticipate and balance the demands of its virtual population to improve the experience.

VIII. CONCLUSION

In this paper, we have evaluated how many popular techniques for player modeling in games could translate into the MMORPG genre and how effective they would be at addressing many of the inherent challenges that the genre brings with it. In addition, we have provided some evidence, by use of a case study, that some of these techniques show great promise if game designers choose to use them in a MMORPG. This case study also shows how our desiderata can be applied to evaluate the effectiveness of player modeling techniques in a MMORPG environment.

In the future, we would like to see research on applying some of these techniques to MMORPGs. The MMORPG genre has been relatively unexplored as far as player modeling is concerned, and it is doubtful that a better source of player observations exists in the game field. By focusing research efforts in this area, we would not only be advancing our understanding of player modeling, but we would be providing much needed innovation to a genre that seems to currently be plagued with imitation.

TABLE 2. Summary of technique performance on desiderata.

Technique	Scalability	New Data	Authorial Burden	Unsupervised Performance	Noise Tolerance	Accuracy
Manual Tagging	●	○	○	○	○	◐
Memory-Based CF	○	●	●	●	◐	●
Model-Based CF	●	◐	●	●	◐	●
Planning-Based Goal Recognition	○	◐	○	●	○	◐
Probabilistic Goal Recognition	●	◐	◐	◐	●	●

APPENDIX
SUMMARY OF RATINGS

Here we provide a summary of the ratings for each technique discussed in terms of the desiderata that we defined. This summary is shown in Table 2.

REFERENCES

[1] S. J. Gross and C. M. Niman, "Attitude-behavior consistency: A review," *Public Opinion Quart.*, vol. 39, no. 3, pp. 358–368, 1975.

[2] B. Harrison and D. L. Roberts, "Using sequential observations to model and predict player behavior," in *Proc. 6th Int. Conf. Found. Digital Games*, 2011, pp. 91–98.

[3] C. Lewis and N. Wardrip-Fruin, "Mining game statistics from web services: A World of Warcraft armory case study," in *Proc. 5th Int. Conf. Found. Digital Games*, 2010, pp. 100–107.

[4] R. Bartle, "Hearts, clubs, diamonds, spades: Players who suit MUDs," *J. MUD Res.*, vol. 1, no. 1, p. 19, 1996.

[5] C. M. Bateman and R. Boon, *21st Century Game Design*. Hingham, MA, USA: River Media, 2006.

[6] I. B. Myers, M. H. McCaulley, and R. Most, *Manual: A Guide to the Development and Use of the Myers-Briggs Type Indicator*. Palo Alto, CA, USA: Consulting Psychologists Press, 1985.

[7] R. Houlette, "Player modeling for adaptive games," in *AI Game Programming Wisdom 2*, S. Rabin, Ed. Stamford, CT, USA: Cengage Learn., 2004, pp. 557–566.

[8] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen, "Interactive storytelling: A player modelling approach," in *Proc. Artif. Intell. Interact. Digital Entertainment Conf.*, Stanford, CA, USA, 2007, pp. 43–48.

[9] R. D. Laws, *Robin's Laws of Good Game Mastering*. Austin, TX, USA: Steve Jackson Games, 2002.

[10] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Adv. Artif. Intell.*, vol. 2009, Aug. 2009, Art. ID 421425.

[11] H. Yu and M. O. Riedl, "A sequential recommendation approach for interactive personalized story generation," in *Proc. 11th Int. Conf. Auto. Agents Multiagent Syst.*, vol. 1. 2012, pp. 71–78.

[12] S. Li and L. Shi, "The recommender system for virtual items in MMORPGs based on a novel collaborative filtering approach," *Int. J. Syst. Sci.*, vol. 45, no. 10, pp. 2100–2115, 2013.

[13] N. ThaiSon and L. Siemon, "Impact of sequence mining on webpage recommendations in an access-log-driven recommender system," Free Univ. Bolzano, Bolzano, Italy, Tech. Rep., 2012.

[14] L. Si and R. Jin, "Flexible mixture model for collaborative filtering," in *Proc. Int. Conf. Mach. Learn. Workshop*, vol. 3. 2003, pp. 704–711.

[15] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[16] N. Silver, "Introducing pecota," in *Baseball Prospectus 2003*. Sterling, VA, USA: Potomac Books, 2003, pp. 507–514.

[17] M. Sharma, M. Mehta, S. Ontanón, and A. Ram, "Player modeling evaluation for interactive fiction," in *Proc. AIIDE Workshop Optim. Player Satisfaction*, 2007, pp. 19–24.

[18] P. Hingston, C. B. Congdon, and G. Kendall, "Mobile games with intelligence: A killer application?" in *Proc. IEEE Conf. Comput. Intell. Games (CIG)*, Aug. 2013, pp. 1–7.

[19] S. Wess, K.-D. Althoff, and G. Derwand, "Using *k*-d trees to improve the retrieval step in case-based reasoning," in *Topics in Case-Based Reasoning*. Berlin, Germany: Springer-Verlag, 1994, pp. 167–181.

[20] D. DeCoste, "Collaborative prediction using ensembles of maximum margin matrix factorizations," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 249–256.

[21] K. Yu, X. Xu, J. Tao, M. Ester, and H.-P. Kriegel, "Instance selection techniques for memory-based collaborative filtering," in *Proc. 2nd SIAM Int. Conf. Data Mining (SDM)*, 2002, p. 16.

[22] K. Miyahara and M. J. Pazzani, "Improvement of collaborative filtering with the simple Bayesian classifier," *Inf. Process. Soc. Jpn. J.*, vol. 43, no. 11, pp. 3429–3437, 2002.

[23] D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency networks for inference, collaborative filtering, and data visualization," *J. Mach. Learn. Res.*, vol. 1, pp. 49–75, Sep. 2001.

[24] S. H. S. Chee, J. Han, and K. Wang, "RecTree: An efficient collaborative filtering method," in *Data Warehousing and Knowledge Discovery*. Berlin, Germany: Springer-Verlag, 2001, pp. 141–151.

[25] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 16. 1999, pp. 688–693.

[26] G. Shani, D. Heckerman, and R. I. Brafman, "An MDP-based recommender system," *J. Mach. Learn. Res.*, vol. 6, no. 2, pp. 1265–1295, 2006.

[27] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proc. 14th Conf. Uncertainty Artif. Intell.*, 1998, pp. 43–52.

[28] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: Using social and content-based information in recommendation," in *Proc. Nat. Conf. Artif. Intell.*, 1998, pp. 714–720.

[29] A. Zook, S. Lee-Urban, M. R. Drinkwater, and M. Riedl, "Skill-based mission generation: A data-driven temporal player modeling approach," in *Proc. 3rd Workshop Procedural Content Generat. Games*, 2012, p. 6.

[30] A. E. Zook and M. O. Riedl, "A temporal data-driven player model for dynamic difficulty adjustment," in *Proc. Conf. Artif. Intell. Interact. Digital Entertainment*, 2012.

[31] H. Yu and M. O. Riedl, "Data-driven personalized drama management," in *Proc. 9th AAAI Conf. Artif. Intell. Interact. Digital Entertainment*, 2013, pp. 191–197.

[32] W. Min, J. P. Rowe, B. W. Mott, and J. C. Lester, "Personalizing embedded assessment sequences in narrative-centered learning environments: A collaborative filtering approach," in *Artificial Intelligence in Education*. Berlin, Germany: Springer-Verlag, 2013, pp. 369–378.

[33] H. A. Kautz, "A formal theory of plan recognition," Ph.D. dissertation, Bell Lab., Murray Hill, NJ, USA, 1987.

[34] S. Carberry, "Techniques for plan recognition," *User Model. User-Adapted Interact.*, vol. 11, nos. 1–2, pp. 31–48, 2001.

[35] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea, "Machine recognition of human activities: A survey," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1473–1488, Nov. 2008.

[36] E. Y. Ha, J. P. Rowe, B. W. Mott, and J. C. Lester, "Goal recognition with Markov logic networks for player-adaptive games," in *Proc. 7th Artif. Intell. Interact. Digital Entertainment Conf.*, 2011, pp. 32–39.

[37] R. Wilensky, "Why John married Mary: Understanding stories involving recurring goals," *Cognit. Sci.*, vol. 2, no. 3, pp. 235–266, 1978.

[38] J. F. Allen and C. R. Perrault, "Analyzing intention in utterances," *Artif. Intell.*, vol. 15, no. 3, pp. 143–178, 1980.

[39] M. Ramírez and H. Geffner, "Probabilistic plan recognition using off-the-shelf classical planners," in *Proc. Conf. Amer. Assoc. Artif. Intell. (AAAI)*, 2010, pp. 1121–1126.

[40] M. Riedl, C. J. Saretto, and R. M. Young, "Managing interaction between users and agents in a multi-agent storytelling environment," in *Proc. 2nd Int. Joint Conf. Auto. Agents Multiagent Syst.*, 2003, pp. 741–748.

[41] J. Harris and R. M. Young, "Proactive mediation in plan-based narrative environments," in *Intelligent Virtual Agents*. Berlin, Germany: Springer-Verlag, 2005, pp. 292–304.

[42] E. Charniak and R. P. Goldman, "A Bayesian model of plan recognition," *Artif. Intell.*, vol. 64, no. 1, pp. 53–79, 1993.

[43] H. H. Bui, "A general model for online probabilistic plan recognition," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 18. 2003, pp. 1309–1318.

[44] N. Blaylock and J. Allen, "Corpus-based, statistical goal recognition," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 18. 2003, pp. 1303–1308.

[45] D. W. Albrecht, I. Zukerman, and A. E. Nicholson, "Bayesian models for keyhole plan recognition in an adventure game," *User Model. User-Adapt. Interact.*, vol. 8, nos. 1–2, pp. 5–47, 1998.

[46] B. Mott, S. Lee, and J. Lester, "Probabilistic goal recognition in interactive narrative environments," in *Proc. 21st Nat. Conf. Artif. Intell.*, vol. 1. 2006, pp. 187–192.

[47] K. Gold, "Training goal recognition online from low-level inputs in an action-adventure game," in *Proc. 6th Int. Conf. Found. Digital Games*, 2010, pp. 21–26.

[48] J. Orkin, T. Smith, H. Reckman, and D. Roy, "Semi-automatic task recognition for interactive narratives with eat & run," in *Proc. 3rd Intell. Narrative Technol. Workshop*, 2010.

[49] E. Y. Ha, J. P. Rowe, B. W. Mott, and J. C. Lester, "Recognizing player goals in open-ended digital games with Markov logic networks," in *Plan, Activity and Intent Recognition: Theory and Practice*, G. Sukthankar, R. Goldman, C. Geib, D. Pynadath, and H. H. Bui, Eds. San Mateo, CA, USA: Morgan Kaufmann, 2014.

[50] C. W. Geib and R. P. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars," *Artif. Intell.*, vol. 173, no. 11, pp. 1101–1132, 2009.

[51] D. H. Hu and Q. Yang, "CIGAR: Concurrent and interleaving goal and activity recognition," in *Proc. 23rd Nat. Conf. Artif. Intell.*, 2008, pp. 1363–1368.

[52] N. Lesh, "Adaptive goal recognition," in *Proc. Int. Joint Conf. Artif. Intell.*, vol. 15. 1997, pp. 1208–1214.

[53] N. Lesh, "Scalable and adaptive goal recognition," Ph.D. dissertation, Univ. Washington, Seattle, WA, USA, 1998.

[54] K. Vanlehn *et al.*, "The andes physics tutoring system: Lessons learned," *Int. J. Artif. Intell. Edu.*, vol. 15, no. 3, pp. 147–204, 2005.

[55] G. N. Yannakakis, "Game AI revisited," in *Proc. 9th Conf. Comput. Frontiers*, 2012, pp. 285–292.

[56] E. Tomai and R. Salazar, "Simulating adaptive quests for increased player impact in MMORPGs," in *Proc. 8th Artif. Intell. Interact. Digital Entertainment Conf.*, 2012, pp. 185–190.

STEPHEN G. WARE received the B.S. degree in computer science and philosophy from Loyola University, New Orleans, LA, USA, in 2008, and the M.S. and Ph.D. degrees in computer science from North Carolina State University, Raleigh, NC, USA, in 2011 and 2014, respectively. He is currently an Assistant Professor of Computer Science with the University of New Orleans, New Orleans. His research interests are in artificial intelligence and computational models of narrative, in particular, automated planning, knowledge representation, narratology, cognitive science, and their applications in games and other virtual environments. He is a member of the Association for Computing Machinery and the Association for the Advancement of Artificial Intelligence.

MATTHEW W. FENDT received the B.S. degree in computer science from the University of Delaware, Newark, DE, USA, in 2009, and the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA, in 2014. He is currently a Lecturer of Computer Science with Baylor University, Waco, TX, USA. His research interests are creating suspense in computer narrative and player behavior and modeling.

BRENT HARRISON received the B.S. degree in computer science and the B.A. degree in English from Auburn University, Auburn, AL, USA, in 2008, and the M.S. and Ph.D. degrees in computer science from North Carolina State University, Raleigh, NC, USA, in 2012 and 2014, respectively. He is currently a Research Scientist with the Entertainment Intelligence Laboratory, Georgia Institute of Technology, Atlanta, GA, USA. His research interests are in machine learning, player modeling, and artificial intelligence for adaptive games. He is a member of the Association for the Advancement of Artificial Intelligence.

DAVID L. ROBERTS received the B.A. degree in computer science and mathematics from Colgate University, Hamilton, NY, USA, in 2003, and the Ph.D. degree in computer science from the College of Computing, Georgia Institute of Technology, Atlanta, GA, USA, in 2010. He is currently an Assistant Professor of Computer Science with North Carolina State University, Raleigh, NC, USA. His research interests lie at the intersection of machine learning, social and behavioral psychology, and human–computer interaction. He has a particular focus on computation as a tool to provide insight into human behavior in narrative, virtual world, and game environments. He is a member of the Association for the Advancement of Artificial Intelligence and the Association for Computing Machinery.