Analytics-Driven Dynamic Game Adaption for Player Retention in a 2-Dimensional Adventure Game

Brent Harrison and David L. Roberts
North Carolina State University
Raleigh, North Carolina 27606

Abstract

This paper shows how game analytics can be used to dynamically adapt a casual, 2-D adventure game named *Sidequest: The Game* (SQ:TG) in order to increase session-level retention. Our technique involves using game analytics to create an abstracted *game analytic space* to make the problem tractable. We then model player retention in this space and move through this space in accordance to a target distribution of game states in order to influence player behavior. Experiments performed show that the adaptive version of SQ:TG is able to better fit a target distribution of game states while also significantly reducing the quitting rate compared to the non-adaptive version of the game.

Introduction

As casual games continue to grow in popularity, the importance of understanding player *retention* grows immensely. The term retention has many definitions, but in games the retention rate is often used to refer to the percentage of players that continue to play a game after a certain period of time. Player retention is important to game designers for several reasons. It can be used to determine how successful a product was or how successful future products may be. It could be used to determine the health of a community in the case of multiplayer games. For casual and social games, retention is especially important because almost all of their revenue off of a game comes from in-game purchases (often referred to as *microtransactions*) or from in-game advertisements.

There has been much work done on predicting player retention and identifying factors that contribute to retention; however, there has been relatively little work done on actually using these types of models to *influence* retention. In this work, we examine how dynamic game adaption can be paired with game analytics to increase *session-level retention* in a casual game environment. We define session-level retention rate as the percentage of players that complete a single game session. Session-level retention is important in casual game environments as many rely on players completing all available tasks or levels for a given time period and offering the player the ability to purchase additional ones.

Figure 1: A screenshot of Sidequest: The Game

To study how dynamic game adaption can be used to influence session level-retention, we have created a 2-D adventure game named *Sidequest: The Game* (SQ:TG), which serves as a test environment. SQ:TG is a casual, 2-D adventure game where the player must explore the game world and complete quests for NPCs in order to progress through the game. This game mimics other casual games in that it does not take very long to complete (on average a playthrough takes 25 minutes) and there is no real penalty for quitting the game (other than losing progress).

In order to influence player retention, our work leverages the idea that there are game states that are associated with session-level retention and others that are associated with players never finishing the game. Using this, our technique targets a distribution of game states that are predictive of session-level retention while actively avoiding states that are likely to lead to the player quitting the game. SQ:TG, however, is a complex game environment with many possible game states, which makes the problem of explicitly modeling this environment intractable. In the past, we have solved this problem by using a set of game analytics to abstract the game state into a smaller *game analytic space* (Harrison and Roberts 2013). Using this strategy, it becomes possible adapt game environments by altering the analytic values that are used to describe the game analytic space.

Related Work

To date, most of the research done on player retention in games has focused on long-term retention over several

months or years. Here, we will review the relevant literature on player retention.

A great deal of the research on player retention has targeted retention in massively-multiplayer online role-playing-games (MMORPGs). This is not unexpected because most of these games have monthly subscription fees, which means that player retention has a direct influence on the total revenue for the game. Also, people can play these games for several years, making them an ideal environment to study long-term retention. There have been several studies that explore the possible factors that contribute to retention in these environments. These factors can range from in-game actions (Tarng, Chen, and Huang 2008; 2009), demographic information (Debeauvais et al. 2011; 2014), or even player motivation (Borbora et al. 2011). Recently, there has been work that uses a player's social network as a basis for retention prediction (Kawale, Pal, and Srivastava 2009).

There has also been work done on modeling player-retention in other types of game environments. Weber, Mateas, and Jhala used regression to determine what features most contribute to retention in both Madden '11 (2011) and Infinite Mario (2011).

In the realm of social and casual games, Lin *et al.* (2013) studied players' motivations for play in social games and found that progression, not social interaction, was often the most important factor in players continuing to play. Contrary to this finding, there has been work done that shows that the existence of a well-defined game community can have a noticeable influence on player retention (Kuo et al. 2009). There has also been work documenting how secondary objectives in casual games could lead to a drop in player retention (Andersen et al. 2011), contrary to popular wisdom.

There has also been much research done on creating adaptive games and game personalization. Zook *et al.* (2012) use tensor factorization to predict player performance in order to recommend missions to players. Spronck *et al.* (2004) use dynamic scripting to adjust difculty by affecting how AI chooses rules to follow. Shaker *et al.* (2010) use neural networks to automatically generate levels for Infinite Mario.

In this work we focus on how an adaptive system can be used to *influence* retention. In previous work (Harrison and Roberts 2013), we found that dynamic game adaptions can be used to influence session-level retention in a game environment designed to mimic *Scrabble*. In this work, we build on our previous work and see how a similar technique performs in a much more complex game environment.

Sidequest: The Game

Sidequest: The Game (see Figure 1), is a 2-dimensional adventure game coded in Flash in which the player takes control of a hero with the goal of becoming an adventurer. The hero is free to explore the world and is able to talk to friendly non-player characters (NPCs) to receive quests. The goal of the game is to complete three game *stages* by completing 3 quests in each stage. During each stage of the game, different quests are made available to the player. Each stage contains 10 unique quests which are randomly distributed to NPCs throughout the world. In total, there are 30 possible quests for the player to complete. Once the player has finished the 3 quests that are required to advance to the next stage, it is not possible to accept any other quests. This means that a player that completes the game will finish a total of 9 quests.

Although there are 30 unique quests in the game, there are only limited number of quest types to complete. These quest types included quests that involved killing some number of enemy NPCs, quests that involved talking to certain NPCs in other areas of the world, and quests that involved solving puzzles or riddles in the game.

Throughout the course of the game, the player can accept any number of possible quests, but they can only have one active quest at a time. If a player wants to change quests, they need only abandon their current one by accepting a different one from a different quest-giver. Players are also free to reject any quests that do not sound appealing based on the description. This was done to give the player the freedom to perform the types of tasks that they enjoyed and still give us an idea of what specific goal they are working toward.

The game logs several low-level and high-level features about gameplay. These features include information on the quests that a player accepts/rejects/completes/abandons, the number of enemies defeated, the NPCs that the player interacted with, and how close a each quest-giving NPC is to the player at any given time.

Methodology

In previous work (Harrison and Roberts 2013), we outlined a process for increasing session-level retention involving game analytics. The first step in this process is abstracting the full space of possible game states into a manageable set of states described by a set game analytics and then modeling session-level retention in this space. In practice, this is done by defining a set of *vanity analytics* with which to describe the game state. Vanity analytics are analytics that typically have a great deal of descriptive and predictive power but are difficult to directly manipulate. In games, this is often due to the fact that these analytics directly describe player behavior, and player behavior is difficult to directly manipulate. An example of vanity analytics include the number of enemies a player has killed or the specific quests a player has completed. Since it is difficult to directly control these values, they are vanity analytics. The reason that the game analytic space is usually defined in terms of vanity analytics is because the session-level retention models are constructed in this space, meaning that vanity analytics are often used to increase the accuracy of these models.

The second step involves dynamically adapting the game world by moving through this space in accordance to a target distribution of game states by altering the values of a set of *actionable analytics*. Actionable analytics are difficult to use in creating predictive models, but it is possible to directly control their value. They are often not used in modeling because they can take on several possible values, thus increasing the number of observations needed to describe them exponentially. Actionable analytics also do not generally consider player actions, which makes them especially unappealing for modeling any type of player behavior. That being said, their ability to be directly manipulated makes

them ideal for implementing dynamic game adaptions since it is clear how to affect actionable analytics. An example of an actionable analytic is the location of health packs in a first-person shooter. Since it is easy to control exactly where health packs are located, it is an actioanble analytic. This difference between vanity and actionable analytics indicates that a gap exists between actionable and vanity analytics that must be bridged in order to create dynamic game adaptions. In the following sections, we will describe how we perform each of these steps in SQ:TG in greater detail.

Game State Abstraction

The first step of our dynamic game adaption scheme is to abstract the game state using a set of vanity analytics. Since accepting and completing quests is the primary game mechanic in SQ:TG, we only use analytics that describe how players interact with quests to create the game analytic space. In this work, we use the following 4 possible quest interactions to describe the game analytic space: accepting a quest, rejecting a quest, abandoning a quest, and completing a quest. For this work, we chose only to consider *how* a player interacts with a quest rather than including information on *which* quest the player was interacting with. The reason for this is that we wanted to mitigate the effects of the curse of dimensionality by reducing the number of analytics that we consider. As this number grows, the number of observations we need to sufficiently describe the space of analytic value combinations grows exponentially. By only considering a small number of analytics, we reduce the number of observations needed to describe the space. Using this scheme, the entire game state can be described as the current player's sequence of quest interactions.

n-Gram Models of Retention

As with previous work (Harrison and Roberts 2013), we choose to use n-grams to model session-level retention in the game analytic space. n-gram modeling consists of making a Markov assumption that the event only depends on the previous $n-1$ events. This means that we do not need to consider the player's entire event history when predicting session-level retention, which is ideal since the amount of observations required to model the full space of event histories would grow exponentially as their length increased. Using this, it is possible to calculate the probability that a player will quit the game by calculating:

$$P(c|i,s_n) = \frac{P(i|c)P(s_n|c)P(c)}{P(i,s_n)} \quad (1)$$

This equation is used to calculate the probability that a player quits the game (c is the player's class describing whether they quit the game or not) given their most recent n events, s_n, on turn i. As mentioned earlier, the movement through the game analytic space that results in game adaption occurs in accordance to a target distribution of game states. This is generated using a set of analytic sequences that are predictive of the player quitting the game. To find these sequences, we use Equation 1 to calculate that probability for every observed n-event sequence on each *turn* of the game. In SQ:TG, we consider a turn to start when the player accepts their first quest of a stage and end when the player completes the final quest necessary to advance to the next stage. If $P(c|i,s_n)$ is greater than the *a priori* probability of correctly predicting that a player will quit, then we consider that sequence predictive of players quitting on that turn.

Table 1: The percentage of the game spent in a warning state for event sequences of length 2 and 3.

Length	Complete	Incomplete	Difference
2-Events	22.2%	47.0%	24.8%
3-Events	20.3%	43.2%	22.9%

Table 2: Goal transition matrix for the second stage of SQ:TG. *Acc* refers to accepting a quest, *Rej* refers to rejecting a quest, *Aba* refers to abandoning a quest, and *Com* refers to completing a quest.

		Action 2			
		Acc	Rej	Aba	Com
Action 1	Acc	0.03	0.33	0.0	0.64
	Rej	0.0	0.0	0.0	1.0
	Aba	0.0	0.0	0.0	0.0
	Com	1.0	0.0	0.0	0.0

This model contains one free parameter: the number of events, n, that are contained in a sequence. To select this value, we performed a user study in which we deployed SQ:TG online and asked people to play through the game. When data collection was complete, we had collected 266 game traces, of which 141 were completed games and 125 were incomplete. We used this data to calculate $P(c|i,s_n)$ for $n=2$ and $n=3$ to determine which one has more discriminative power. We only consider these values because they both provide a balance between accurately describing the player's event history (defined by how many events we consider) and mitigating the effects of the curse of dimensionality by reducing the number of observations required to fully model the space of possible sequences. We define discriminative power based on the percentage of game time that players spend in a *warning state*. A player is said to be in *warning state* if their last n observed events are predictive of them quitting the game. Ideally, we want to observe players that finish the game spending less time in a warning state than those who do not finish the game. As such, we define discriminative power as follows:

$$d_n = w_i - w_c \quad (2)$$

In other words, we calculate the discriminative power associated with a sequence length, d_n, by subtracting the percentage of time that players who completed the game spent in a warning state, w_c, from the percentage of time that players who did not complete the game spent in a warning state, w_i. The results of this study are shown in Table 1. As seen in the table, players who did not finish the game spent a greater percentage of the game in a warning state than those who did complete the game. This difference was statistically signif-

icant ($p < 0.05$) for both values of n according to a two-tailed, independent samples T-test. As such, we a bigram ($n = 2$) model of player actions to model player retention.

Once this was done, we extracted the list of predictive sequences for each turn and used them to generate a set of 3 target distributions using Markov chain Monte-Carlo (MCMC) sampling. We then turned these dsitributions into transition matrices in order to easily generate goal action sequences. An example transition matrix is shown in Table 2. According to this transition matrix, a player that accepts a quest should complete it most of the time (64%); however, there is a chance that the player should reject a quest or accept a different quest instead of completing their active quest (33% for rejecting a quest and 3% for accepting another quest). Once a transition matrix has been made for each stage of the game, it can be used to create goal states necessary to dynamically adapt the game environment.

Dynamic Game Adaption

In order to move through the analytic game space, we have to first find a way to express game states in terms of actionable analytics. Since the game analytic space and the retention models are defined using vanity analytics, it is impossible to directly control them. This means that in order to dynamically adapt SQ:TG, we must first bridge the gap between the vanity analytics used describe session-level retention, and the actionable analytics that the game will alter to make the adaptions.

In SQ:TG, we use quest proximity to the player as the actionable analytic that will be altered to create game adaptions. In SQ:TG, the AI does not have control over the position of quest-giving NPCs; however, it does have control over the quests that NPCs can give out. Whenever the player enters a screen that contains quest-giving NPCs, the AI will probabilistically assign quests to all visible quest-giving NPCs. By altering how the AI distributes these quests, we can control the proximity of a quest to the player relative to other quests on the screen. Using this strategy, our adaptive system can move through the game analytic space in order to bring about the desired game state.

At a high level, our technique for dynamic game adaption in SQ:TG contains the following steps:

1. Generate a goal analytic state using the transition matrix for the current stage
2. Return the set of example game states that most resemble the current game state
3. Determine a candidate set of quests to place based on the goal analytic state
4. Assign quests to quest-givers such that quests that are likely to move the current state closer to the goal state are closer to the player

Each of these steps will be discussed in greater detail below using an informative example.

Generate Goal State Before any adaptions can be made, a goal must be generated from the target distribution of game states. This involves using the transition matrices that were generated using our target distributions to generate a sequence of events that we will target. Our system uses the transition matrix to generate a sequence of actions that begins with the player accepting a quest and ends with them completing a quest. The resulting sequence is the *target game sequence*. In our example, let's consider the target game sequence of *Accept, Reject, Complete* generated by the transition matrix described in Table 2.

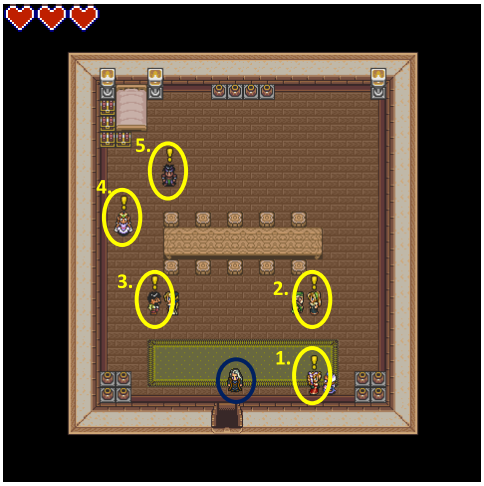


Figure 2: A screenshot of characters in *Sidequest: The Game*. Characters circled in yellow are quest-giving NPCs. Quest-giving NPCs are also numbered 1 through 5. The character circled in blue is the player's character.

Retrieve Example Game States In our example, our target game sequence is *Accept, Reject, Complete*. This means our system needs to distribute quests to NPCs such that it is likely that this event sequence will occur. As mentioned previously, the gap between actionable and vanity analytics must be bridged before any dynamic game adaptions can be made. For modeling retention, we only used events such as a player accepting or rejecting a quest. For creating the adaptions to SQ:TG, we must alter how close *specific quests* are to the player at any given time. In order to bridge the gap between these two types of analytics, we must first determine which quests that a player is likely to accept, complete, reject, or abandon so that they can be intelligently placed around the game world.

This is done by extracting game states similar to the current one from a corpus of observed games. Specifically, our system finds the k-nearest neighbors (with $k = 5$, chosen arbitrarily) to the player's current game state and uses those to extract the set of quests that the current player is likely to accept, complete, abandon, and reject. Similarity is calculated by summing the number of common quest interactions between game states. For example, if game state 1 and game state 2 only shared one quest interaction, such as accepting quest 1, then their similarity value would be 1. The higher the similarity value, the more similar two game states are. For the remainder of this example, we will refer to these game states as *candidate game states*.

Determine Candidate Quests to Place After extracting this set of similar game states, they must be used to determine how to assign quests to quest-givers in the game world in order to bring about the target sequence of events. For this example, let's assume that the player enters the screen in Figure 2. In this figure, there are 5 quest-givers (circled in yellow and numbered from 1-5). The player's character is circled in blue. Since this screen contains 5 quest-givers, the system needs to place 5 quests. The types of quests that the system places completely depends on the goal sequence. Since the target game sequence in this example is *Accept, Reject, Complete* the system needs to place quests that the player will likely accept and eventually complete (since they do not abandon the quest or accept another quest before completing their active quest) as well as quests that the player will reject. The set of candidate quests to place are generated by examining the candidate game states gathered earlier. In this case, the set of candidate quests to place would be each *valid* quest that was either completed or rejected in the candidate game states. A valid quest is a quest that the player has not interacted with in the current game. This is because quests that the player has interacted with are locked to the current quest-giving NPC and cannot be assigned to a different NPC.

Assign Quests to Quest-Givers Once our system has generated the set of candidate quests, it only needs to determine the best action to take. This means that our system has to select quests and assign them to quest-givers such that it is likely that the player will transition into the goal state (as is defined by the target game sequence generated earlier). The action that moves the current game state closer to the goal state can be thought of as placing a quest in the world such that it is likely that it is accepted, rejected, abandoned, or completed based on what is needed. In our example, we need the player to accept a quest that they will eventually complete after first rejecting a second quest. As such, the system would place a quest that the player is likely to complete such that the proximity in terms of Euclidean distance between its quest-giver and the player at the time of placement is minimized. In this case that would mean giving quest-giver 1 pictured in Figure 2 a quest that the player is likely to complete. The exact quest given is the valid quest that was completed most often in the candidate game states retrieved earlier. The next step is to give the next most proximal quest-giver (quest giver 2 in Figure 2) a quest that the player is likely to reject (since the next action to be completed in the goal sequence is to reject a quest). This process is performed just as it was for placing a quest that the player is likely to complete. This process repeats until all quest-giving NPCs on the screen have a quest associated with them.

Evaluation

To validate our technique for increasing session-level retention in SQ:TG through dynamic game adaption, we ran a set of experiments and tested how well our system was able to accomplish two goals: fit a target distribution of analytic game states and reduce the quitting rate in SQ:TG. As such, we use two metrics to determine how successful our system was at accomplishing these goals. First, we measure the Jensen-Shannon divergence (Lin 1991) between the target distribution and the observed distribution of player actions. Second, we measure the *quitting rate*, the percentage of players that did not finish the game. For these experiments, we compare the adaptive version of SQ:TG with a non-adaptive version to determine how effective our system is at increasing session-level retention as well as influencing player behavior.

Data Collection

For these analyses, we performed two separate data collections. During the first one (which was briefly discussed in previously), players could only play the non-adaptive version of the game. As mentioned previously, we collected 266 game traces from 263 players (constituting a 1.1% replay rate) during this data collection. Of these, 141 were complete games and 125 were incomplete. This data was used to generate the n-gram models of session-level retention that are used to generate the target distributions. This data is used as our baseline for evaluation and will be referred to as the *baseline data* for the rest of the paper.

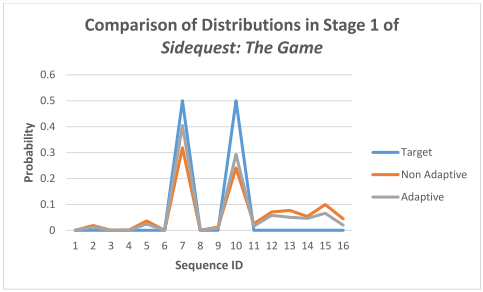
Once this data collection was complete, we used it to create an adaptive version of SQ:TG and performed a second data collection. During this round of data collection, all players played the adaptive version of SQ:TG. By the end of this round of data collection, we had gathered 138 game traces from 138 unique players (constituting a 0% replay rate). This data will be referred to as the *adaptive data* for the remainder of the paper.

Distribution Analysis

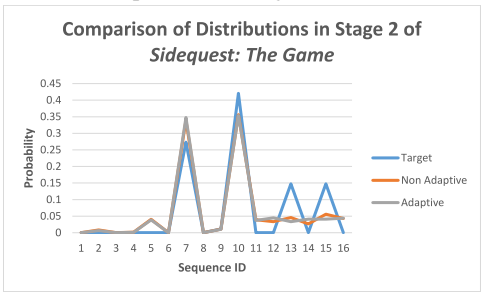
The first analysis that we performed was to determine if our adaptive version of SQ:TG was able to better fit a targeted distribution of player behavior than the non-adaptive version. Figure 3 shows a visual comparison of the analytic distributions produced by the adaptive and non-adaptive versions of SQ:TG to the target behavior distribution. Visual inspection indicates that the adaptive version of SQ:TG was able to better fit the target distribution for Stages 1 and 3 (as evidenced by how well the peaks of the dsitribution are fit), whereas it is unclear which version of SQ:TG performs better in Stage 2 of the game.

To provide more definitive evidence, we calculated the Jensen-Shannon divergence between the target distribution of game analytics and the observed distributions of game analytics produced by both the adaptive and non-adaptive versions of SQ:TG. Jensen-Shannon divergence is a popular method of measuring the similarity between two probability distributions. At a high level, it measures how much of the entropy present in the target distribution is unexplained by the test distribution. As such, the Jensen-Shannon divergence between two distributions will be zero only when the two distributions are identical.

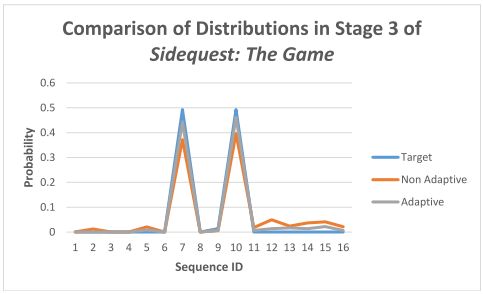
The results of this analysis are summarized in Table 3. As hinted at by the visual inspection, the adaptive version of SQ:TG is able to outperform the non-adaptive version in Stages 1 and 3; however, the non-adaptive version of SQ:TG was able to outperform the adaptive version during Stage 2



(a) Distribution Comparison for Stage 1 of *Sidequest: The Game*



(b) Distribution Comparison for Stage 2 of *Sidequest: The Game*



(c) Distribution Comparison for Stage 3 of *Sidequest: The Game*

Figure 3: Comparing the target distribution to the distributions created by the adaptive and non-adaptive versions of *Sidequest: The Game* in each stage. Sequence ID refers to number associated with a particular 2-event sequence. Probability refers to the probability that a specific 2-event sequence was observed.

of the game. This indicates that our our adaptions were able to influence behavior in 2 out of the 3 stages of the game.

Quitting Rate Analysis

To determine how well the adaptive version of SQ:TG was able to retain players, we analyzed the quitting rate of both versions of the game. Table 4 summarizes the result of this analysis. As seen in the table, the adaptive version of SQ:TG achieved a quitting rate of 34.1%, which is much lower than the 47.0% quitting rate achieved by the non-adaptive version of the game. We used Fisher's exact test to measure the significance of this difference and found that this difference was significant with $p = 0.015$.

To give these results some context, we also analyzed what percentage of players quit during each stage of the game. Those results are shown in Table 5. As the table shows, a smaller percentage of players quit the adaptive version of SQ:TG than the non-adaptive version during each stage of the game. What is interesting to note, however, is how similar the quit rates were during Stage 2 for both versions of the game. This difference will be discussed in greater detail in the next section.

Table 3: Jensen-Shannon divergence values comparing the distributions created by the adaptive/non-adaptive version of *Sidequest: The Game* and the target distribution.

	Stage 1	Stage 2	Stage 3
Adaptive	0.12	0.11	0.03
Non-Adaptive	0.19	0.09	0.09

Table 4: Comparison between the non-adaptive and adaptive versions of *Sidequest: The Game* in terms of finished and unfinished games. Also given is the percentage of total games that were unfinished.

	Finished	Unfinished	Percentage
Adaptive	91	47	34.1%
Non-Adaptive	141	125	47.0%

Discussion

First, it is important to note that in terms of Jensen-Shannon divergence, the adaptive version of SQ:TG better fits the target distribution in 2 out of the 3 stages of the game. This leads us to believe that this technique is fully capable of affecting player behavior in this environment under the right circumstances. It is interesting to note, however, that our system was *not* able to perform this task during the second stage of the game. This was shown to be true during the distribution analysis as well as the analysis of players that quit during each stage of the game. One possible explanation for this is that the target distribution for Stage 2 was much more complicated than the target distributions for Stages 1 and 3. If you examine the visual representation of the target distributions in Figure 3 you'll see that the target distributions for Stages 1 and 3 have 2 peaks whereas the target distribution for Stage 2 has 4 peaks.

Despite this, we still feel that there is enough evidence to say that this technique for implementing dynamic game adaptions to increase session-level retention is effective in this environment. We showed that we could bridge the gap between vanity and actionable analytics to create dynamic game adaptions that could influence player behavior. We also showed that this shift in behavior coincided with a significant drop in the quitting rate, which, while not definitive,

Table 5: The percentage of players that quit at each stage of *Sidequest: The Game*

	Stage 1	Stage 2	Stage 3
Adaptive	27.5%	6.6%	0.0%
Non-Adaptive	35.3%	7.2%	3.7%

certainly gives evidence to the notion that this change in behavior influenced the drop in the quitting rate.

Future Work

The most important avenue of future work is to explore the limitations of this technique. It is not clear how the complexity of the target distribution affects this technique's performance. The ability to identify what properties a target distribution needs to exhibit for this technique to be successful would be useful for others that wish to use this technique.

This technique has, to this point, been tested only in casual game environments. It would be worthwhile to see how this technique generalized to other game types. One possible line of research involves seeing how this technique performs in larger, more complex games where game sessions are not clearly defined. This would require the use of analytics that are powerful enough to incorporate more information about a player's history while still retaining the ability to sufficiently simplify the representation of the game state.

Conclusion

In this paper we show how an adaptive system can leverage the descriptive and prescriptive power of game analytics to improve session-level retention in *Sidequest: The game*. This system uses n-gram models of session-level retention to identify game states in the game analytic space that are predictive of players quitting the game. The system then intelligently places quests throughout the world in order to induce a target distribution of analytic states that are predictive of player retention. Our results show that the adaptive version of SQ:TG is able to better fit a target distribution of game states than its non-adaptive counterpart. We also provide evidence that this shift in behavior results in a reduction in the quitting rate, thus increasing session-level rate.

This work serves as evidence to the power that game analytics have for more than just being purely descriptive or predictive tools in games. We have shown that they can be used to dynamically make changes to game environments in order to influence player behavior. We hope that this work will encourage others to discover new ways that game analytics can be used not just to describe or predict behaviors, but to dynamically create and shape new player experiences.

References

Andersen, E.; Liu, Y.; Snider, R.; Szeto, R.; Cooper, S.; and Popovic, Z. 2011. On the harmfulness of secondary game objectives. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, 30–37.

Borbora, Z.; Srivastava, J.; Hsu, K.-W.; and Williams, D. 2011. Churn prediction in mmorpgs using player motivation theories and an ensemble approach. In *Privacy, Security, Risk and Trust (passat), Third International Conference on Social Computing (socialcom)*, 157–164. IEEE.

Debeauvais, T.; Nardi, B.; Schiano, D. J.; Ducheneaut, N.; and Yee, N. 2011. If you build it they might stay: Retention mechanisms in World of Warcraft. In *Proceedings of the 6th International Conference on Foundations of Digital Games*, 180–187. ACM.

Debeauvais, T.; Lopes, C. V.; Yee, N.; and Ducheneaut, N. 2014. Retention and progression: Seven months in World of Warcraft. In *Proceedings of the 9th International Conference on Foundations of Digital Games*.

Harrison, B., and Roberts, D. L. 2013. Analytics-driven dynamic game adaption for player retention in scrabble. In *Computational Intelligence in Games (CIG), 2013 IEEE Conference on*, 1–8. IEEE.

Kawale, J.; Pal, A.; and Srivastava, J. 2009. Churn prediction in MMORPGs: A social influence based approach. In *International Conference on Computational Science and Engineering*, 423–428. IEEE.

Kuo, Y.-l.; Lee, J.-C.; Chiang, K.-y.; Wang, R.; Shen, E.; Chan, C.-w.; and Hsu, J. Y.-j. 2009. Community-based game design: experiments on social games for commonsense data collection. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, 15–22. ACM.

Lin, Z.; Lewis, C.; Kurniawan, S.; and Whitehead, J. 2013. Why players start and stop playing a chinese social network game. *Journal of Gaming & Virtual Worlds* 5(3):307–328.

Lin, J. 1991. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on* 37(1):145–151.

Shaker, N.; Yannakakis, G. N.; and Togelius, J. 2010. Towards automatic personalized content generation for platform games. In *AIIDE*.

Spronck, P.; Sprinkhuizen-Kuyper, I.; and Postma, E. 2004. Difficulty scaling of game ai. In *Proceedings of the 5th International Conference on Intelligent Games and Simulation (GAME-ON 2004)*, 33–37.

Tarng, P.; Chen, K.; and Huang, P. 2008. An analysis of WoW players' game hours. In *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, 47–52. ACM.

Tarng, P.-Y.; Chen, K.-T.; and Huang, P. 2009. On prophesying online gamer departure. In *8th Annual Workshop on Network and Systems Support for Games (NetGames), 2009*, 1 –2.

Weber, B.; John, M.; Mateas, M.; and Jhala, A. 2011. Modeling player retention in Madden NFL 11. In *Twenty-Third Annual Conference on Innovative Applications of Artificial Intelligence (IAAI)*.

Weber, B.; Mateas, M.; and Jhala, A. 2011. Using data mining to model player experience. In *Foundations of Digital Games Workshop on Evaluating Player Experience in Games*.

Zook, A., and Riedl, M. O. 2012. A temporal data-driven player model for dynamic difficulty adjustment. In *AIIDE*.