HARRISON, BRENT E. Dynamically Adapting Games Using Vanity and Actionable Analytics to Increase Session-Level Retention. (Under the direction of David Roberts.)

The ability to control player retention has been long sought after by both game designers and game researchers. With casual games becoming more popular, it has become increasingly important to have a good understanding of how to influence player retention. In the game industry, there are many accepted rules about how to best maximize player retention in a game environment (Fields, 2014). These rules are based on expert knowledge and were created based on years of experience making games, but have not been explicitly validated. In academia, many researchers have studied what contributes or detracts from player retention in various game environments with varying amounts of success. Typically, those studying retention choose to examine how well games can retain players over the entire lifetime of the game. In my work, I examine the problem of session-level retention.

Given the recent rise in popularity of social and casual games, I believe that learning to influence session-level retention can be just as useful as learning to influence long-term retention. In this dissertation, I propose an analytics-driven technique for increasing session-level retention through the use of dynamic game adaption. This technique leverages two different types of analytics, vanity analytics and actionable analytics, to create models of player retention and then use them to dynamically alter game worlds to increase session-level retention. Vanity analytics are those that hold a great deal of predictive power but are difficult to directly affect, whereas actionable analytics are difficult to directly use in modeling player behavior but can be manipulated in real time.

This technique offers several benefits to both game designers as well as game researchers. Since this technique is data-driven, the insights gained are grounded in data which gives them more strength than some of the insights gained from expert knowledge. Also, this technique uses the inherent strengths of both vanity and actionable analytics and provides researchers with a method for incorporating models of player retention into game environments.

My research will proceed in three phases. First, I will use data-driven techniques to create computational models of session-level retention using sets of vanity analytics. Then, I will use these models to determine how to adapt two different game worlds in an attempt to influence session-level retention. Finally, these adaptions will be evaluated based on
how well they actually do influence session-level retention.

In addition, I evaluate the side-effects that dynamic game adaption has on player experience to make sure that any gains in session-level retention do not come at the cost of player experience. During this evaluation, I measure the effects of dynamic game adaption on two measures of play experience, intrinsic motivation and player engagement.
Dynamically Adapting Games Using Vanity and Actionable Analytics to Increase Session-Level Retention

by
Brent E. Harrison

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Computer Science

Raleigh, North Carolina
2014

APPROVED BY:

Michael Young

James Lester

Robert St. Amant

Magy Seif El-Nasr

David Roberts
Chair of Advisory Committee
DEDICATION

To my parents.
BIOGRAPHY

Brent Harrison's interest in research began simply enough. While studying to complete his undergraduate degree at Auburn University, Brent was given the opportunity to take part in undergraduate research. There was but one condition on this offer: he had to implement a genetic algorithm to solve the traveling salesman problem and then watch it converge. Watching this program slowly converge towards the optimum solution, Brent realized that his plans for the future had changed.

This simple act led him to consider research as a career. Once he graduated from Auburn University with an undergraduate degree in Computer Science and English, he came to North Carolina State University. In 2010, Brent began working with Dr. David Roberts studying data-driven player modeling and how they could be used to create adaptive games.
ACKNOWLEDGEMENTS

First, I would like to thank each member of my committee: Michael Young, James Lester, Rob St. Amant, and Magy Seif El-Nasr. Each of you has helped me immensely through your questions and comments on my work.

I would also like to thank my advisor, David Roberts, for being helping me grow as a researcher and as a person, one unplanned meeting at a time. I would also like to thank my first advisor, Nagiza Samatova. You taught me the value of hard work and what it truly means to be a researcher.

During my time at NC State, I have had many friends and colleagues that have helped me on my way. Zach Jorgensen and Trisha Biswas helped me get through my first teaching assistantship as a graduate student for a class that none of us had ever taken. Grad school was never very scary after that experience. It is difficult to name everyone that I’ve had the pleasure to work with, laugh with, and converse with, but I am sure going to try: Julio Bahamon, Rogelio Cardona-Rivera, Brad Cassell, Justus Robertson, Jerry Yang, Robert Loftin, Titus Barik, Matt Fendt, John Rowe, Alok Baikadi, Lucy Shores, and a cast of others. I would also like to acknowledge my best friend, Stephen Ware. You gave me a lab when I had none, you gave me an advisor when I had none, and you moved my furniture when I was stranded in Tennessee. I couldn’t ask for anything more out of a best friend.

I would also like to thank my parents and family for constantly supporting me, even when it seemed as though I would be a student forever.

Finally, I would like to thank Dr. Hari Narayanan and Dr. Gerry Dozier, my undergraduate research advisors. You both showed me that the world was a much bigger place than I thought it was. For that, I am eternally grateful.
# TABLE OF CONTENTS

LIST OF TABLES ................................................................. ix

LIST OF FIGURES ............................................................. xii

Chapter 1 Introduction ...................................................... 1
  1.1 Introduction ............................................................... 1
  1.2 Motivation ................................................................. 2
    1.2.1 Data-Driven Game Adaption .................................... 3
    1.2.2 The Gap Between Actionable And Vanity Analytics ......... 4
    1.2.3 Session-Level Retention ......................................... 6
  1.3 Summary of Dissertation .............................................. 6
    1.3.1 n-gram Models for Session-Level Retention ................. 8
    1.3.2 DDBIAG .............................................................. 8
    1.3.3 Quantitative Evaluation ....................................... 9
    1.3.4 Psychometric Side-Effects .................................... 9
  1.4 Overview of Game Environments .................................... 10
    1.4.1 Scrabblesque ...................................................... 10
    1.4.2 Sidequest: The Game .......................................... 11
  1.5 Conclusion .............................................................. 14

Chapter 2 Background and Related Work ................................ 15
  2.1 Introduction ............................................................. 15
  2.2 Modeling ................................................................. 15
    2.2.1 Modeling Retention .............................................. 16
      2.2.1.1 In Games .................................................... 16
      2.2.1.2 Outside of Games ......................................... 19
    2.2.2 Modeling Behavior .............................................. 20
      2.2.2.1 Modeling Behavior in Games ............................ 21
      2.2.2.2 Modeling Behavior Outside of Games .................. 25
    2.2.3 n-Gram Modeling ............................................... 27
      2.2.3.1 n-Gram Modeling in Games .............................. 27
      2.2.3.2 n-Gram Modeling Outside of Games .................... 28
    2.2.4 Knowledge Modeling ............................................ 30
  2.3 Adaptive Systems ...................................................... 31
    2.3.1 Adaptive Systems in Games .................................. 31
    2.3.2 Adaptive Systems Outside of Games ........................ 33
  2.4 Background on Psychometrics ...................................... 36
    2.4.1 Intrinsic Motivation ......................................... 36
    2.4.2 Engagement ...................................................... 37
<table>
<thead>
<tr>
<th>Chapter 3</th>
<th>Actionable and Vanity Analytics</th>
<th>39</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>39</td>
</tr>
<tr>
<td>3.2</td>
<td>Actionable and Vanity Analytics</td>
<td>39</td>
</tr>
<tr>
<td>3.3</td>
<td>Analytics in Games</td>
<td>40</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Actionable Analytics in Games</td>
<td>41</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Vanity Analytics in Games</td>
<td>41</td>
</tr>
<tr>
<td>3.4</td>
<td>The Gap Between Analytics</td>
<td>42</td>
</tr>
<tr>
<td>3.5</td>
<td>Game Environments</td>
<td>44</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Motivation</td>
<td>44</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Vanity Analytics and Actionable Analytics in Scrabblesque</td>
<td>45</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Vanity Analytics and Actionable Analytics in Sidequest: The Game</td>
<td>47</td>
</tr>
<tr>
<td>3.6</td>
<td>Conclusion</td>
<td>49</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 4</th>
<th>Predicting Session-Level Retention</th>
<th>51</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>51</td>
</tr>
<tr>
<td>4.2</td>
<td>Raw Vanity Analytics</td>
<td>52</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Modeling Session-Level Retention in Scrabblesque</td>
<td>52</td>
</tr>
<tr>
<td>4.2.1.1</td>
<td>Data Collection and Experimental Methodology</td>
<td>53</td>
</tr>
<tr>
<td>4.2.1.2</td>
<td>Results and Discussion</td>
<td>55</td>
</tr>
<tr>
<td>4.3</td>
<td>n-Gram Modeling Using Deviation-Based Analytics</td>
<td>56</td>
</tr>
<tr>
<td>4.3.1</td>
<td>SAX Transformation Methodology</td>
<td>57</td>
</tr>
<tr>
<td>4.3.2</td>
<td>n-Gram Modeling</td>
<td>58</td>
</tr>
<tr>
<td>4.4</td>
<td>Prediction in Scrabblesque</td>
<td>59</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Deviation-Based Analytics in Scrabblesque</td>
<td>59</td>
</tr>
<tr>
<td>4.4.2</td>
<td>The n-Gram Model in Scrabblesque</td>
<td>60</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Experimental Methodology</td>
<td>61</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Results and Discussion</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>Prediction in SQ:TG</td>
<td>65</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Analytics in Sidequest: The Game</td>
<td>66</td>
</tr>
<tr>
<td>4.5.2</td>
<td>The n-Gram Model in Sidequest: The Game</td>
<td>67</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Experimental Methodology</td>
<td>68</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Results and Discussion</td>
<td>70</td>
</tr>
<tr>
<td>4.6</td>
<td>Conclusion</td>
<td>72</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>Bridging the Gap</th>
<th>73</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>73</td>
</tr>
<tr>
<td>5.2</td>
<td>Motivation</td>
<td>74</td>
</tr>
<tr>
<td>5.3</td>
<td>The DDBIAG Algorithm</td>
<td>75</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Select a Goal State</td>
<td>76</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Retrieve Game States</td>
<td>76</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Generate Candidate Actions and Resultant States</td>
<td>77</td>
</tr>
</tbody>
</table>
Chapter 7  Related Work and Methodology

7.3  Analysis of Tactic-Based Engagement

7.3.1  Predictive Models of Engagement

7.3.2  Targeted Engagement

7.3.3  Analysis of the Aspects of Engagement

7.4  Discussion

7.5  Psychometric Evaluation in Sidequest: The Game

7.5.1  Methodology and Data Collection

7.5.2  Analysis of Intrinsic Motivation and Engagement

7.5.3  Analysis of the Aspects of Engagement

7.5.4  Discussion

7.6  Conclusion

Chapter 8  Conclusions

8.1  Summary of Work

8.2  Limitations

8.3  Impact

8.4  Future Work

8.5  Concluding Remarks

BIBLIOGRAPHY
LIST OF TABLES

Table 4.1  Summary of Scrabblesque dataset. Games played shows the number of games played in each category. Game length reports the average game length and standard deviation in game turns. Players shows the number of unique players in each category. .......................... 54

Table 4.2  Prediction accuracies for predicting session-level retention on each turn in Scrabblesque. After the turn 6 the prediction accuracy never increases. Values that are greater than the prediction threshold are bolded. .............. 56

Table 4.3  Average percentage of a game spent in warning state for each feature in Scrabblesque. ................................................................. 62

Table 4.4  Number of sequences with $P(c|i, s) > 0.241$ in the beginning, middle, and end of a game of Scrabblesque. Beginning is the first five turns, middle is six through ten, and end is 11 and on. ........................................ 63

Table 4.5  Percentage of values at each position in informative action sequences for each feature. Position 1 represents the earliest event looked at while Position 3 represents the most recent event looked at. ................................. 65

Table 4.6  Summary of Sidequest: The Game dataset. Games played shows the number of games played in each category. Players shows the number of unique players in each category. .................................................. 69

Table 4.7  Percentage of time spent in warning state when completing a quest marks the end of a turn across all conditions. Rows in bold indicate that differences found in this condition were statistically significant ($p < 0.05$) according to a two-tailed, independent samples T-Test. ............... 70

Table 4.8  Percentage of time spent in warning state when completing a stage marks the end of a turn across all conditions. Rows in bold indicate that differences found in this condition were statistically significant ($p < 0.05$) according to a two-tailed, independent samples T-Test. ............... 70

Table 4.9  The number of action sequences that are predictive of players quitting during each stage of Sidequest: The Game ................................. 71

Table 4.10 Percentage of time that an action occurs in a predictive sequence for each act. Percentages are calculated as the number of times an action did appear divided by the number of times that it could appear. ............... 72

Table 5.1 Transition Matrices for Sidequest: The Game ................................. 84

Table 6.1 KL divergence values. Shows divergence between the adaptive/non-adaptive distributions and the target distribution (Direction 1) and between the target distribution and the adaptive/non-adaptive distributions (Direction 2). The average for each algorithm is also shown. ........................................ 93
Table 6.2 | Comparison between the non-adaptive version of *Scrabblesque* and the adaptive version of *Scrabblesque* using the DDBIAG algorithm in terms of the number of finished and unfinished games. | 93
---|---|---
Table 6.3 | Number of times that players exhibited the desired behavior for each analytic as well as the total number of attempts to elicit this behavior in players. | 94
Table 6.4 | $K_L$-divergence values after marginalizing the words submitted analytic. Since $K_L$-divergence values differ depending on the order in which a comparison is made, I averaged the results across both directions. Direction 1 refers to the comparison of the adaptive/non-adaptive distribution to the target distribution whereas Direction 2 refers to the comparison of the target distribution to the adaptive/non-adaptive distribution. | 96
Table 6.5 | Comparison between the non-adaptive version *Scrabblesque* and the adaptive version of *Scrabblesque* using the DDBIAG algorithm in terms of the number of finished and unfinished games. Updated to reflect the inclusion of additional data gathered from an additional round of data collection. | 96
Table 6.6 | $K_L$-divergence values after marginalizing the words submitted feature. Updated to reflect the inclusion additional data gathered from an additional round of data collection. | 96
Table 6.7 | The percentage of games that ended in a low, medium, or high number of turns. | 97
Table 6.8 | The percentage of players that played multiple games of *Scrabblesque*. | 98
Table 6.9 | Jensen-Shannon divergence values comparing the distributions created by the adaptive/non-adaptive version of *Sidequest: The Game* and the target distribution. | 102
Table 6.10 | 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. | 103
Table 6.11 | The percentage of players that completed each stage of *Sidequest: The Game* in the adaptive and non-adaptive versions. | 103
Table 6.12 | The percentage of players that quit at each stage of *Sidequest: The Game*. | 103
Table 6.13 | The percentage of players that quit at each stage of *Sidequest: The Game*. | 104
Table 7.1 | The Intrinsic Motivation Inventory. Responses are measured on a 7-point Likert scale with 1 meaning *not true* and 7 meaning *very true*. | 109
Table 7.2 | The Intrinsic Motivation Inventory. Responses are measured using a 3-point Likert scale with 1 meaning disagreement, 2 meaning neither agreement or disagreement, and 3 meaning agreement. | 110
Table 7.3 | Summary of IMI results in *Scrabblesque*. | 111
Table 7.4 | Summary of GEQ results in *Scrabblesque*. | 111
Table 7.5  T-Test Results for the comparison of the non-adaptive version of Scrabblesque against the version of Scrabblesque with the DDBIAG algorithm. Results are given for both the IMI and the GEQ ......................... 112
Table 7.6  Summary of data and T-Test Results for GEQ Subscales in Scrabblesque ........ 113
Table 7.7  Summary of IMI results in Sidequest: The Game ......................... 114
Table 7.8  Summary of GEQ results in Sidequest: The Game ......................... 114
Table 7.9  T-Test Results for the comparison of the non-adaptive version of Sidequest: The Game against the adaptive version of Sidequest: The Game. Results are given for both the IMI and the GEQ ......................... 115
Table 7.10 Summary of data and T-Test Results for GEQ Subscales in Sidequest: The Game ................................................................. 116
LIST OF FIGURES

Figure 1.1 A screenshot of Scrabblesque .......................... 3
Figure 1.2 A screenshot of Sidequest: The Game .................. 4
Figure 1.3 A visualization of my dissertation. Boxes (excluding the game world) indicate major contributions. Chapter annotations are given to direct the reader to the appropriate chapter for a contribution. The green area refers to parts of adaption that do not occur during gameplay. The yellow area refers to parts of adaption that happen during gameplay. ........... 7

Figure 3.1 A heatmap showing player deaths in a level of Half-Life 2 ................. 43
Figure 3.2 A screenshot of Scrabblesque. The game board has been outlined in red and the player's rack of letter tiles has been outlined in blue. ............... 45
Figure 3.3 Quest-givers in Sidequest: The Game. For ease of understanding, characters that have quests to offer the player are circled in yellow. The player is circled in blue. .......................... 48

Figure 4.1 Probability the player will quit playing Scrabblesque based on the score difference feature. The short dotted line is the baseline, the solid line is the probability of the player eventually quitting at each turn for an example finished game, and the long dotted line is the probability of the player eventually quitting at each turn for an example unfinished game. 62
Figure 4.2 Turn histograms for the word submitted and word length analytics in Scrabblesque. This shows how many predictive sequences were found on each turn. .......................... 64

Figure 5.1 A flowchart showing the execution of my algorithm. First, a goal is chosen. Then, instances of this goal are found in the knowledge database and used to evaluate a set of candidate actions. ......................... 76
Figure 5.2 Target Distributions of Actions for Each Stage in Sidequest: The Game .. 83
Figure 6.1 Direct comparisons between the target distribution and the adaptive distribution using my algorithm and non-adaptive distribution. Each graph shows probabilities for each possible feature combination. The adaptive distribution appears to be shifted slightly, but overall matches the target distribution more closely than the non-adaptive distribution. 92
Figure 6.2 Direct comparisons between the target distribution and the adaptive distribution using my algorithm and the non-adaptive distribution after the word submitted analytic has been marginalized out. Each graph shows probabilities for each possible feature combination. ........... 95
Figure 6.3  Replicated comparison between the target distribution and the adaptive distribution using my algorithm and the non-adaptive distribution after the *word submitted* analytic has been marginalized out with additional data. .................................................. 98

Figure 6.4  Comparing the target distribution to the distributions created by the adaptive and non-adaptive versions of *Sidequest: The Game*. ............. 101

Figure A.1  A DBN representing student knowledge over time. Boxes with dashed outlines indicate hidden nodes in the graph while boxes with solid outlines represent observable nodes in the graph. ................................. 138

Figure A.2  A DBN showing the initial student knowledge node added in. In the model put forth by Pardos and Heffernan, there would be an initial knowledge node for each student that could be modeled. ................. 140

Figure A.3  An example of using matrix factorization for student performance prediction. ................................................................. 144
1.1 Introduction

The goal of my research is to show how data-driven techniques can be used to dynamically adapt game content to bring about certain player behaviors in casual games. In particular, I am interested in increasing session-level player retention. Session-level player retention refers to the number or percentage of players that successfully complete a game session. Based on this goal, the thesis statement for this dissertation is as follows:

Thesis  Data-driven techniques can be used to bridge the gap between the predictive power of vanity analytics and the prescriptive power of actionable analytics to dynamically adapt game experiences in order to increase session-level retention in a variety of casual game environments through altering analytic values.

At a high level, vanity analytics are those that are used to describe the state of the player or of the environment which are, by definition, difficult or impossible to directly manipulate. Actionable analytics, however, are those analytics that describe the aspects of the game environment that are directly under my control. These analytics are typically very useful
for modeling, but they are often difficult to directly influence. From this thesis statement, I have determined that there are two questions that I must answer.

1. How can I incorporate both vanity analytics and actionable analytics into a dynamic game adaption technique that manages to bridge the gap between these analytics?

2. Will this technique be powerful enough to increase session-level player retention in different casual game environments?

In order to answer these questions, my work proceeds in three phases. During the first phase of research, I use a set of vanity analytics to create predictive models of session-level retention. During the second phase of research, these models are incorporated into a dynamic game adaption algorithm to create adaptive user experiences. During this phase of work, the fundamental gap that exists between vanity analytics and actionable analytics must be bridged in order to generate the adaptive experiences that I desire. At a high level, this is done by dynamically altering a set of analytics that are associated with session-level retention. The final phase of my research involves determining the quality of my game adaption algorithm by quantitatively evaluating session-level player retention in multiple game environments.

To address external validity, I have chosen to study two different game environments of varying complexity. The first is a simple game environment based on the board game Scrabble, which I have named Scrabblesque (Figure 1.1). The second, named Sidequest: The Game (SQ:TG) is a game environment in which the player completes various quests in order to advance to the end of the game (Figure 1.2). I have chosen these two environments because they represent two very different, yet very common, game genres: casual puzzle games and casual adventure games. They also present varying degrees of complexity with respect to AI that the user plays against, the amount of player interaction available, and the amount of control I have over the game world. By examining these two game environments, I show how well my technique generalizes to different game genres.

### 1.2 Motivation

There are several motivations for various aspects of this work. In the following sections, I will discuss the most important of these in detail.
1.2.1 Data-Driven Game Adaption

Recent research has made it clear that there are different aspects of gameplay that appeal to different players (Bartle, 1996). Therefore, game designers create games that include different types of gameplay to appeal to a wide audience. An effect of this type of design is that games can become quite large and contain content that some players may never see.

In response to this, there has been an increasing interest in *adaptive* video games. An adaptive video game is a video game that changes itself based on the player that is currently playing it. For example, an adaptive game may present the player with more combat opportunities if it determines that the current player prefers to fight enemies. Examples of adaptive games include games that alter the difficulty of the game to produce challenging experiences (typically through increasing the skill of the enemies present) and games that guide the player towards content that they will enjoy (usually by having NPCs verbally guide the player to this content). Current techniques to create adaptive game environments focus on using player self-report data (such as surveys or interviews) (Sharma et al., 2010) and knowledge engineering (such as expert or author knowledge) (Thue et al., 2008) to create these environments and adaptions; however, these techniques have inherent, conceptual flaws that could negatively impact performance. The use of self-report data, for
example, has been shown to have little correlation with actual behavior (Gross and Niman, 1975). As for the use of knowledge engineering, the quality of a model or adaptive game environment is completely dependent on the quality of the knowledge used to build this model. If the creator of the model or the environment is using flawed knowledge during the creation process, then the quality of the product will suffer.

I feel that the use of data-driven techniques addresses these issues by simply removing the human element from the problem. By using data-driven techniques, I ensure that the behaviors that I model are grounded in the data that players actually produced. In addition, data-driven techniques can help motivate the types of adaptations that need to be made as they will arise organically from player data. Currently, there has been little work done on using data to create adaptive game environments. The goal of this work is to show that data-driven algorithms can be used during both behavior model creation and actual game adaption to create an adaptive game environment that increases session-level retention. To accomplish this goal, I first need to bridge the gap between actionable analytics and vanity analytics.

1.2.2 The Gap Between Actionable And Vanity Analytics

In previous sections, I have given a high level definition of both actionable and vanity analytics. The reason that these analytics are so important is because they give access to two important types of information that are, unfortunately, mutually exclusive much of the
time. Vanity analytics have a large amount of predictive power; however, they often cannot be directly affected. This is usually because vanity analytics deal with the intangibles of a system. In terms of game environments, the greatest intangible of all is the player and which behaviors the player will exhibit. This means that often times researchers will use vanity analytics to construct computational models of player behavior, which is understandable since they can provide excellent insights into the workings of player behavior. The problem with this is that it is difficult to use these models since, in practice, their analytics are very rarely actionable. Actionable analytics are those that are directly affectable by an entity such as the game designer, developer, or in my case, an autonomous AI agent. Actionable analytics, contrary to vanity analytics, are more difficult to use in modeling player behaviors since they often do not capture the context in which the player is acting.

This fundamental difference between actionable and vanity analytics leads to a gap in knowledge that needs to be bridged in order to create a data-driven, dynamic game adaption algorithm. To better illustrate this gap, I will use an example grounded in American politics. In this example, consider a politician running for office. When a politician is running for office, public opinion polls are often used to determine how well the candidate is performing. In general, these polls can be used to predict the outcome of an election as the candidate that is rated higher in the polls usually wins. The candidate, however, cannot directly manipulate the value of the polls. There is no direct cause and effect relationship between any action available to the politician and their standing in the polls. What the politician can do, however, is distribute money in an attempt to indirectly influence the polls. In this example, the polls are a vanity analytic describing the current state of the election. The polls have a great amount of predictive power, but they contain very little prescriptive power since it is unclear what actions need to be taken in order to directly affect poll values. How and where the politician spends her money, however, is an example of an actionable analytic. This is not necessarily a good predictor of the outcome of the election, but it is something that the politician has complete control over.

In order to generate high-quality game adaptions, this gap needs to be bridged. In other words, a strategy needs to be developed for turning the predictive insights contained in vanity analytics into clear changes that can be made to actionable analytics.
1.2.3 Session-Level Retention

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 produces 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 (Fields, 2013).

Typically, game designers attempt to optimize long-term retention. Long-term retention typically involves getting players to play a game over several months or even several years. There has been much work on what affects long-term retention in games and what aspects of game design can influence long-term retention. In this dissertation, I choose not to focus on long-term retention in favor of exploring what I call session-level retention. Session-level retention rate refers to the percentage of players that complete a game session. Most casual games have a clear beginning and a clear end to a game session. In puzzle-games, completing a level or a set of levels may constitute the end of a game session. In an adventure game, completing the list of available quests for the day could constitute the end of a game session. In casual games that are more open ended, such as Farmville, a game session could be defined as performing all actions that are allowed before leaving the game. In casual games, maximizing the number of players that complete a game session is especially important since many casual games offer players a chance to extend game sessions through the use of microtransactions, meaning that players have to make it to the end of a game session to take advantage of this type of offer.

1.3 Summary of Dissertation

In Figure 1.3, I have given a visualization of my dissertation work. This visualization gives an overview of what the process for creating an adaptive game environment is. At a high level, the first step of creating adaptive game environments consists of taking information from the game world and then using this data to create a model of player behavior. This process, an offline process, is done through the analysis of various vanity analytics. Then,
Figure 1.3 A visualization of my dissertation. Boxes (excluding the game world) indicate major contributions. Chapter annotations are given to direct the reader to the appropriate chapter for a contribution. The green area refers to parts of adaption that do not occur during gameplay. The yellow area refers to parts of adaption that happen during gameplay.
these models are used as inputs into an experience management system (DDBIAG in the figure), which is used (in combination with data received from the game world), to make changes to the game world by altering the values of actionable analytics that describe said game world.

Using this figure as a guide, I will now go over each part of my dissertation in greater detail.

### 1.3.1 n-gram Models for Session-Level Retention

Recall that the first step in creating an adaptive game environment is to create a model of player behavior. During this step, I leverage the predictive power of vanity analytics to create models of player behavior.

For this, I use an \( n \)-gram model to calculate the probability that players will quit the game early. \( n \)-gram models assume that the current observation (in this case, an observation would be a player action) depends only on the previous \( n - 1 \) actions. These models have been used frequently in text mining and classification. More recently, they have been shown to be effective in predicting goals in interactive narrative environments (Mott et al., 2006). Due to their success in these other fields, I choose to use \( n \)-grams to model session-level player retention. In particular, I am interested in determining what specific action sequences are associated with session-level retention. A more in depth discussion of this model is given in Chapter 4.

### 1.3.2 DDBIAG

Once the player models have been created, they are then used by an experience manager (Riedl et al., 2008) to alter the game environment to enhance player experience. An *experience manager* is an autonomous agent that uses a model of player behavior in order to make alterations to the game environment. The experience manager makes use of an algorithm named *data-driven backwards induction for action generation* (DDBIAG) in order to generate the adaptations made in a game environment. In order to generate these adaptations, the DDBIAG algorithm needs to use the player models generated using vanity analytics to determine how to alter a set of actionable analytics. By changing the values of these actionable analytics, the DDBIAG algorithm actually changes the game environment in an attempt to *indirectly* bring about session-level retention.
At a high level, the DDBIAG algorithm uses *backwards induction* (Von Neumann and Morgenstern, 2007) in order to select adaptations. Backwards induction is a technique for selecting the optimal sequence of actions to reach a goal state by selecting the action that should be performed immediately preceding the goal state. How this translates into game adaption is that my system chooses the action that moves the game world closest to the intermediate goal state chosen by the experience manager. This system is discussed in detail in Chapter 5.

### 1.3.3 Quantitative Evaluation

Each piece of work that I have discussed up until this point will be subject to a quantitative evaluation. The evaluation of the player models that I have created (Chapter 4) will evaluate the predictive power of my models. The evaluation of the DDBIAG algorithm (found in Chapter 6) will evaluate whether or not it is able to influence player behavior and influence session-level retention. In this evaluation, I will be comparing the behaviors observed in a game environment that has been augmented with my experience manager against a game environment with no experience management system present.

### 1.3.4 Psychometric Side-Effects

While the primary goal of the DDBIAG algorithm is to increase session-level retention, it is important to consider the possible side-effects that this system could have. Ideally, my adaptive game environments should increase session-level retention without adversely affecting the play experience. During this evaluation, I measure the effect that the DDBIAG algorithm had on two psychometrics describing player experience. The first of these is *intrinsic motivation*. Intrinsic motivation is defined as motivation based on the inherent satisfactions derived from action (Ryan and Deci, 2000). To test this, I will use a subset of the questions available from the intrinsic motivation inventory (IMI).

The second psychometric that I want to study is engagement. The game engagement questionnaire (Brockmyer et al., 2009) is a questionnaire that measures several aspects of player engagement (such as flow or presence) in order to create a unified idea of a player's engagement during a game. Using this questionnaire, I plan to measure the differences in player engagement across all experimental groups.

To reiterate, the goal of this analysis is to determine what, if any, effect the DDBIAG
algorithm had on these psychometrics describing player experience. While it would be ideal if the DDBIAG algorithm was able to somehow increase both intrinsic motivation and player engagement, it will be sufficient if the DDBIAG algorithm is able to increase player retention while still maintaining the same level of player engagement and intrinsic motivation present in a non-adaptive version of the same game. More details on this evaluation can be found in Chapter 7.

1.4 Overview of Game Environments

In order to better understand the behavior models that I build and how they are used to create game adaptions, it is important to better understand the game environments that I will be working in and the types of analytics that I will be working with. In the next sections, I will introduce both Scrabblesque and SQ:TG and describe the types of analytics that are gathered during gameplay.

1.4.1 Scrabblesque

Scrabblesque is a Flash game that is a modified version of the popular board game, Scrabble. Scrabblesque simulates a casual game environment by providing players with a limited number of possible actions and a limited amount of ways to interact with the game. It is also relatively simple to begin playing Scrabblesque and quite simple to end the game. The main difference between Scrabblesque and an actual social game is that the player does not play against another human, but rather competes against a computer-controlled AI. Also, the game ends when either the player or the computer-controlled AI reaches 150 points.

Scrabblesque is designed to log several low level features of gameplay. These features are:

- Mouse Presses and Releases: The x and y coordinates of mouse presses or releases
- Player Words and Score: The word that a player played as well as that word’s point value
- Computer Words and Score: The word that the computer played as well as that word’s point value
- Player Word Location: Where on the board the player played a specific word
• Computer Word Location: Where on the board the computer played a specific word
• Player Rack Status: The current tiles in the player's rack of letter tiles
• Game Beginning and End: When the player begins and ends a game as well as that game's outcome

Notice that I am logging a combination of vanity and actionable analytics in this game. Analytics such as the word a player played and its resulting score value are vanity analytics since they are beyond our control but very useful for modeling. That being said, the words that computer played and their score value are actionable analytics since both of these values are under control of the AI, and therefore, subject to any alterations that an experience manager wants to make.

1.4.2 Sidequest: The Game

The second game, Sidequest: The Game (see Figure 1.2), is a 2-dimensional adventure game coded in Flash in which the player takes control of a nameless 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 and can engage in combat with enemies by firing a stream of arrows at them. 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 available to the player. Each stage contains 10 unique quests which are randomly distributed to NPCs throughout the world. In other words, each quest-giver is given a random quest to present to the player, meaning that the experience differs from player to player. In total, there are 30 possible quests for the player to complete. In order to advance to the next stage, the player must complete 3 quests out of the 10 that are possible for their current stage. Once the player has finished 3 quests, 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 7 possible types of quests:

• Kill Quests: Quests requiring the player to kill a certain number of enemies
• Exploration Quests: Quests requiring the player to travel to a certain part of the world and speak to a friendly NPC
• Escort Quests: Quests requiring the player to find a friendly NPC and then bring them back to the quest-giver

• Item Retrieval Quests: Quests requiring the player to retrieve an item in one part of the world and bring it back to the quest-giver

• Item Drop Quests: Quests requiring the player to kill monsters until they drop an item that the player must return to the quest-giver

• Riddle Quests: Quests requiring the player to answer a riddle in order to complete the quest

• Conversation Quests: Quests requiring the player to use knowledge of an NPC’s likes and dislikes to navigate a conversation with them

Throughout the course of the game, the player can accept quests 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 and then accept a different one from a different quest-giver. Players are also free to reject any quests that do not sound appealing to them from the text description of them. 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. In this way we avoid having to solve the goal recognition problem (see Chapter 2). It was also possible for a player to, in some special cases, fail the quest they were on. In the case of riddle quests and conversation quests, it was possible to explicitly fail by selecting incorrect choices. It is possible to fail on every other quest by dying while on a quest. If you fail a quest, it is possible to pick up the same quest again for another chance at completing it.

The game logs several low-level and high-level features about gameplay. These features include:

• Keyboard Presses and Releases: The key that a player presses or releases and the time that it happened

• Quests Accepted: Which quests the player has accepted and when the player accepted them

• Quests Abandoned: The quests that the player accepted but chose not to complete and the time when each quest was abandoned
• Quests Rejected: The quests that were offered to the player but not accepted and the time the player rejected each quest

• Quests Completed: The quests the user finished and when the player finished them

• Deaths: The number of deaths the player experienced as well as where in the world the player died, quest the player was on, and cause of death

• Monster Kills: The number of monsters killed by the player, time each monster was killed, and where each monster was killed

• NPCs Talked To: The NPCs talked to as well as the current quest of the player, the NPC’s location in the world, and the time the player talked to the NPC

• Quest Proximity: The how close each quest-giving NPC is to the player at any given time

All of these analytics are vanity analytics except for Quest Proximity. It is feasible for the AI to control, to some degree, how close certain quests are to the player. As such, this is an actionable analytic.

This game represents a significant step up in complexity from Scrabblesque in many ways. The first of these is that SQ:TG presents the user with a game environment that is much larger than Scrabblesque in terms of virtual space as well as complexity. The user is free to explore the game world as they please (although certain parts of the game world are unavailable to the player until they have reached a certain stage of the game), and they are free to perform quests at their leisure. In addition, there are many different ways that the player can interact with this virtual environment. The player is able to talk to NPCs, fight enemies, as well as (to a limited extent) control the state of the environment through the destruction of certain types of terrain. Also, contrary to Scrabblesque, the AI controls many aspects of the game. The AI is able to control the placement of certain NPCs, the availability and placement of quests, and how enemies behave. This is important because it means that there are more variables available that can be altered to induce different user behaviors.
1.5 Conclusion

In this dissertation, I will show how vanity analytics can be used to model and predict session-level retention in both Scrabblesque and SQ:TG. In addition, I will provide a technique for bridging the gap between vanity and actionable analytics such that the predictive power of vanity analytics can be used by the DDBIAG algorithm to generate dynamic game adaptations. The contributions of this dissertation include:

- Designing algorithms that use vanity analytics to create computational models of session-level retention in games
- Creating a data-driven technique to bridge the gap between vanity and actionable analytics in order to dynamically adapt game environments to increase session-level retention
- A quantitative evaluation of the effectiveness of my techniques at increasing session-level retention in two game environments
- A psychometric evaluation of the side-effects of my dynamic game adaption technique on player experience in two game environments

I believe that this technique for bridging the gap between vanity and actionable analytics will be useful to other researchers, game designers, or other types of developers that want to create model-based adaptive systems.
CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Introduction

This chapter will survey literature relevant to my dissertation research. I begin by discussing topics related to modeling behavior in games. In this section, I review literature relevant to behavior modeling including literature on modeling retention, \( n \)-gram models, and knowledge modeling. I will also review techniques that have been used to adapt/personalize content in different forms of media (specifically in narratives and games). I will conclude with a discussion of background literature for intrinsic motivation and engagement since these are the two psychometrics that I will be measuring in Chapter 7.

2.2 Modeling

In today’s world, understanding how people interact with computational systems is an important task for several reasons. Understanding how people use various systems can help researchers better understand human behavior, which can lead to advances in usability,
security, and other fields. As such, user modeling, creating computational models that describe and predict how users will behave, has been a rich topic for many years. In the following sections, I will review literature that discusses how user modeling has been used both in games and in other fields.

2.2.1 Modeling Retention

As the goal of my research is to develop a technique for increasing session-level retention through dynamic game adaptions, it makes sense to begin with a discussion on the existing work that has been done on retention. Included are discussions of retention research in games and in other areas.

2.2.1.1 In Games

The ability to accurately predict player retention has long been sought after by both games researchers and game designers. As such, there has been a great deal of research on player retention from both game researchers and game designers. That being said, the research that has been done has focused mainly on long-term retention, player retention over a game’s lifetime, whereas my work focuses on session-level retention, player retention over a game session.

Tarng, Chen, and Huang (2008) studied the play tendencies of 34,524 World of Warcraft players during a two year period to determine if there are any indicators that a player would quit playing. They determined that while predicting a player’s short term behavior is feasible, predicting long term behavior is a much harder problem. Tarng, Chen, and Huang (2009) performed a follow-up study on a different MMORPG using support vector machines (SVMs) as their predictor. They concluded that it is possible to predict when a player will quit in the near future as long as they do not possess wildly erratic play behavior.

Researchers have also studied what specific factors contribute to retention. Weber, Mateas, and Jhala used regression to determine what features most contribute to retention in both Madden ’11 (2011a) and Infinite Mario (2011b). In Madden ’11, the authors use three types of regression techniques (linear regression, M5 regression (Wang and Witten, 1997), and additive regression (Friedman, 2002)) to determine the features of gameplay that most influence the number of games that a player will play. Several features were examined, such as offensive play diversity and the number of game changing plays (such
as interceptions). The authors used a similar technique in infinite Mario to rank features based on their influence over player retention.

Similarly, Debeauvais et al. (2011) examined how various demographic factors affected retention in World of Warcraft. In this work, the authors examine how various demographic factors such as age, gender, marital status, and self-identified motivations correlated to the number of hours played per week, whether or not they would take an extended break from the game, and how long they had been playing the game. In addition, they also measured how the social aspects of the game correlated to these retention metrics. While the authors did not specifically seek out to predict retention, the authors did identify several factors that are correlated to long-term retention.

This work was extended in (Debeauvais et al., 2014) where the authors predict retention on a month-by-month basis. In this work, the authors incorporate demographic information, as before, but this time include data on player region (such as China or the United States) as well as progression information. Progression was defined in terms of how far players (or at what rate players) made it through end-game content. They used a simple regression model to predict whether players would quit playing after a month. The results of this study were questionable as the authors achieved very high recall, but very low precision. This means that they predicted that several players that actually kept playing the game would quit.

There has been a recent trend in retention analysis that involves using social networks to aid in prediction. The theory behind these techniques is that players are more likely to quit playing games if other members of their social network quit playing as well. Similarly, if several members of a player’s social network continue to play a game, it is likely that this player will continue as well. Kawale et al. use this theory as the basis for their work in predicting retention in Everquest II (2009). This work consists of using a modified diffusion model (MDM) to determine how the probability that a player will quit affects the probability that players in their social network will quit. This model also incorporates player engagement (defined as the amount of time a player spends in the game) into the calculations done to predict whether a player will quit. Results showed that the authors’ MDM outperformed techniques that used solely social network information or engagement information for predicting retention.

Borbora et al. (2011) compare the use of theory-driven modeling approaches and data-driven modeling approaches for predicting retention in Everquest II. A theory-driven app-
The approach is one that is based on an underlying theory. In this case, that underlying theory is motivation theory. Thus, the authors used individual player motivations to determine if they would continue playing Everquest II or not. In this work, they found that data-driven models often produce more accurate models; however, they claim that the theory-driven models are often much simpler and are easier for domain experts or game designers to understand. This work is significant in that the authors recognize that data-driven models are often difficult to interpret, which can make it difficult to use them as the basis for changes to the game.

In the realm of social and casual games, Lin et al. (2013) studied the reasons that players start and stop playing a Chinese, social network RPG. They used survey data to examine what motivates players to begin playing a social game and what factors influence their decision to either keep playing or quit. What they found was that social interaction was often not a factor in a player’s decision to keep playing. They found that players that kept playing stayed because they enjoyed progressing through the game. It was this sense of achievement that kept them going. Also, they found that boredom was the leading cause of players quitting. This is contrary to the popular belief that social interactions are what drive retention in social games.

Kuo et al. (2009) explored how the difference in game communities can affect player retention. This work examines how two social games that were distributed to different web communities performed in terms of various retention metrics. They found that both games experienced similar retention rates, but mainly differed on the amount of players that played each game. This was most likely due to the fact that one of the games had a preexisting community of players whereas the other had to build the community from the ground up. These results imply that different communities might not have a large impact on retention rates, but a preexisting community can be very important for getting more people to play a game.

Andersen et al. (2011) showed that the presence of secondary objectives can lead to players leaving the game prematurely. They showed this by examining the play times of players when playing a casual game that had secondary objectives (in this case, coins the players could collect) versus players when playing a casual game that had no secondary objectives. They showed that there was a noticeable difference in the amount of players that played the game for long amounts of time.

The work presented here shows that many different techniques have been used to model
and predict player retention across different game genres. Despite these differences, they are alike in that many of the models produced using these techniques are not actionable. This means that the insights gained from these models do not lead to clear actions that need to be taken to influence behavior. For example, social network analysis tells us that players will play longer if people in their social networks play as well. This statement gives game designers a great deal of insight into what motivates people to keep playing a game; however, it does not say how exactly one might get people in their social networks to keep playing. This further reinforces the need for a technique to bridge the gap between vanity and actionable analytics so that the insights made using these types of models can be used to improve retention where necessary.

2.2.1.2 Outside of Games

There has also been much work done on modeling retention in areas outside of games. In 1997, Hennig-Thurau and Klee (1997) put forth the notion that customer retention is a product of customer satisfaction that is moderated by relationship quality. Relationship quality was defined as a 3-dimensional variable that includes the customer's perception of product/service quality, the customer's trust, and the customer's relationship commitment. This was a very early notion of what factors impact retention in a business setting. Theoretically, these insights could apply to any field in which people offer goods or services to others.

Jolley et al. (2006) challenge this view of retention and argue that customer satisfaction is not important if buying habits have already been established. In particular, the authors study how efforts to improve online gambling services effected retention. Their results show that increasing user satisfaction had little effect on customer retention, whereas their frequency of past behavior was a much more influential.

In an effort to further augment the view of customer retention as a function of customer satisfaction by examining how switching barriers affect customer defection. Switching barriers are things that make it difficult to switch from one product to another. Examples include interpersonal relationships, perceived switching costs, and the attractiveness of alternative products. The authors used a survey measuring switching barriers combined with logistic regression to determine that switching barriers along with customer satisfaction was important to explaining customer retention.

Larivière and Van den Poel (2005) use random forests to predict retention (measured in
terms of “next buy” actions) for a large European financial services company. They found that random forests perform better on this data than logistic regression. They also found that past customer behavior as well as demographic information such as age are strong predictors of retention; however, they found that gender as well as geographical information were not as effective at predicting customer retention.

Support vector machines (SVMs) have also been used to predict customer retention in telecommunications. Zhao et al. (2005) use an improved one-class SVM (Schölkopf et al., 2001) to predict customer retention based on demographics, quality of service, and the features of each customer’s subscriber plan. They found that their improved SVM was able to outperform neural networks, decision trees, and naive Bayesian networks on this dataset.

As with retention in games, there has been a recent trend in using social network analysis to predict retention. Dasgupta et al. (2008) examine how social influence can affect retention in mobile phone networks. This is done by using a diffusion-based approach which models customers discontinuing their service as having a set amount of influence over members of their social network. As customers discontinue their service, their influence spreads to others in their social network, which is likely to make them also stop their service. In practice, the authors were able to identify a significant fraction of the customers that were lost in a single month.

While these models and techniques can effectively predict retention in a variety of environments, their insights, as they were in games, are not actionable. The ability to predict losing a customer is half of what is necessary to actually retain that customer. Once they have been identified, steps must be taken to keep people from switching services. Since these models are typically based on customer behavior or demographic information, it is very difficult to translate any insights contained in these models into a clear set of steps that need to be taken to retain customers.

### 2.2.2 Modeling Behavior

There has also been much work done on modeling and predicting user behaviors outside of retention. In the following sections, I will review research on general user modeling both in games and outside of games.
2.2.2.1 Modeling Behavior in Games

One of the earliest attempts to build models of human behavior for games came in 1996 when Richard Bartle (1996) created his four player types (Achievers, Socializers, Explorers, and Killers) by observing player responses to what their favorite parts of a multi-user dungeon (MUD) were. These player types each refer to a different aspect of a game that players could prefer to do. Achievers are players that are concerned with completing objectives explicitly outlined by the game, such as obtaining the best items or reaching a higher level. Socializers mostly play a game in order to interact with other people. Explorers want to see all aspects of the game world. It is important to note that this extends beyond simply exploring the game world map. Explorers often will try to find peculiarities with game mechanics or the game engine itself. Lastly, killers are players that mainly want to disrupt the gameplay experience of other players. These models were based entirely on observational data gathered from the message board associated with the MUD in question. From this observational data, Bartle also proposed how each of these player types interact with one another. For example, increasing the amount of socializers in a game will lead to an increase in the amount of killers in the game.

In 2006, Nick Yee (2006) empirically tested Bartle’s player models by creating a survey based off of these player models. This survey was then given to 3,000 players and then used in a principle components analysis to group the answers. As a result of this PCA, three overall player types emerged: those concerned with achievement, those concerned with socializing, and those concerned with game immersion. The achievement player group is similar to the achievers put forth by Richard Bartle. They are mainly concerned with gaining power and advancement. In addition, this player type encompasses players that are concerned with competition and competing with others. The social player group is also very similar to Bartle’s socializers. The immersion player group contains people that are mainly concerned with discovery and the story of the world around them. It also contains people that are concerned with role-play and the story of their characters.

As more research was done in this area, researchers began to move away from survey-based models into more data-driven models. Lankveld et al. (2009) attempted to create a union of survey-based and data-driven models by trying to directly translate a player personality survey into a game setting. The goal was to use observations of the player as answers to questions that would be on a survey and then build a player model based on these observations. This was done by designing tasks for the player to complete that would
correspond to certain questions or the facets of extroversion that those questions were meant to measure. It is important to note that these models were to be used to determine if in-game actions correlate to player personality. It was shown that this is possible for extroversion. It was also shown that some of the in-game tasks did not correlate with the facets of extroversion that they were meant to measure.

Sharma et al. (2010; 2007) also created a hybrid method by combining game traces along with player survey data in order to create player models using case-based reasoning. In their method, the model creation module monitors a player’s progress through the game and tries to match their progress with a game trace that has already been seen. This is then used to infer how enjoyable the current player will find the game. Once the current player has been matched with a previous player, the current player’s enjoyment is measured by using the previous player’s survey responses. This technique bears resemblance to the theory behind collaborative filtering techniques (Su and Khoshgoftaar, 2009) used in recommender systems.

The previous approaches were based primarily on author observation and player self-report data. My approach to model creation differs from these approaches in that I choose to examine player behavior directly rather than rely on surveys and other self-report data. By using a data-driven approach, I avoid any biases that may be introduced via social desirability or the player simply not being able to accurately describe their own behavior.

Another approach to player modeling is the approach taken by Thue et al. (2008; 2007) in the PaSSAGE system. In this system, actions are annotated as to what type of player is more likely to perform them. For example, an aggressive player is more likely to fight when given the choice. As a player moves through the game, the collection of their actions and the annotations combine to form a player model. The player types used in this system were the ones introduced in Laws’ guide for pen-and-paper role-playing games (Laws, 2001). This approach relies on knowledge engineering as it depends on using predefined annotations to develop their player models. Also, this model assumes that the types developed for a pen-and-paper RPG will translate into a virtual environment.

Another approach by Ryan Houlette (2003) involves creating a hierarchy of actions with basic user actions (such as use smoke bomb) as leaves in the trees and user traits (such as being a stealthy player) as internal nodes. Here, as the player completes basic actions, information about that user’s play style is updated according to the tree’s internal nodes. For example, possible leaf nodes in a tree might correspond to avoiding guards, avoiding
cameras, and moving silently. All of these concrete behaviors would fall under the abstract behavior of *being stealthy*. As the player performs more of these concrete behaviors, their *being stealthy* score will increase. This approach also incorporates knowledge engineering because the tree structure must be known beforehand and is often created using author or expert knowledge.

Some researchers have chosen to create player models that only fit a very narrow scenario. In Brian Magerko and John Laird’s Interactive Drama Architecture (IDA) (2003), player modeling is done with respect to the goals that the story has laid out. The player models are only able to determine if a player will advance to the next plot point within a given amount of time; however it is rather simplistic in its construction because it only uses the current player’s past performance as the basis of this model. This model also uses an internal probabilistic rule-based model to determine the probability that a player will complete a story task (Magerko, 2006). This model must be defined beforehand by the author since it is domain-dependent. Hence, this technique relies on knowledge engineering to be constructed.

The main disadvantage of using these types of techniques is that they require the author of the model to have correct knowledge about how players will behave in a game environment. If this knowledge is incorrect, then the quality of the model can suffer greatly. There has been work done on using observed user models generated in other environments to inform models that are generated using knowledge engineering (Roberts and Roberts, 2011), but it is not yet clear when this technique is the most applicable. As such, I still feel that examining data directly provides the safest means for making quantitative statements about player behavior since they are all based off of behaviors that I have observed.

The methods mentioned previously have been based on small-scale data collection experiments or knowledge engineering. Drachen *et al.* (2009) created player models for *Tomb Raider: Underworld* by examining various statistics gathered from 1365 players. The authors extracted 4 features of gameplay that they used to create these behavior models. The features extracted were

- **Cause of death**: Each death is annotated with what caused it (hostile NPCs, environmental effects (traps), or falling)

- **Total number of deaths**: The total number of times the player died for any reason

- **Completion time**: The time that it takes the player to complete the game in minutes
• Help on demand: The number of times the player uses the in-game help feature for solving puzzles

Their method uses emergent self-organizing maps (ESOMs) (Kohonen, 2001) to cluster the data. It is important to note that the statistics gathered from players were high-level descriptions of play and could not be used to predict individual actions. From their analysis, the extracted 4 player types from the data. The first of these contain players that complete the game very quickly and do not use the in-game help option very often. These players were labeled veterans. The second group extracted contained those players that died quite often and did not finish the game very quickly; however, these players did not ask for in-game help very much. These players were labeled solvers. The third, and largest, player type contains those players that die most frequently from enemies. These people are labeled as pacifists. The final player group, labeled runners, contain players that die most often from enemies or falling, but finish the game very quickly. It is important to note that the labels assigned to these groups actually have no bearing on the behaviors exhibited by these players, and are only done to increase understanding of the groups after the fact.

Shim et al. (2009; 2010; 2010) created models of player behavior for Everquest II using techniques used in home run prediction. Their model is created by examining the leveling speed of a character in the past in order to determine how long until the character reaches the next level. The two algorithms that they use to perform these predictions are PECOTA (Silver, 2003) and MARCEL (Tango, 2004). PECOTA works by finding the nearest neighbor to the current player in order to predict their performance. In other words, it finds players with similar past performances and uses those to make statements about the current player and his or her future performance. The MARCEL algorithm examines the past 3 years of player performance and attempts to extrapolate forward from there. It is important to note that this technique only examines one player at a time, and does not take into account any data gathered from any other players.

The player models used in the EMPath system (Sullivan et al., 2009) use random walks to determine the actions that a user will likely take. This system is similar to the IDA proposed by Magerko and Laird in that the world is designed around plot points that can occur in a story world. The player model is used to determine the probability that the player will perform a certain plot point so that the experience manager present can take actions accordingly. As a result, the world is set up as a decision tree of plot points. In other words, completing certain plot points will open up opportunities to explore different plot points,
while closing off the opportunity to explore others. To create this player model, 100 random story traversals are made to determine which plot points are likely to be performed.

Zook et al. use a tensor factorization technique to predict a player's mastery of a skill in both military training scenarios (2012) and in a game that emulated the combat system used in a turn-based role-playing game (2012). 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 (2012) apply prefix-based collaborative filtering 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.

Li and Shi (2013) use collaborative filtering 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. On a similar note, ThaiSon and Siemon (2013) used collaborative filtering to recommend wiki pages for users to visit based on their play in MMOGs. It is important to note, however, that this method was only used to cluster and make recommendations about websites related to an MMOG 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.

Unlike many of the other data-driven techniques for player modeling, my technique is used to determine which sequences of actions are predictive of certain behaviors. In many other data-driven player models, the end-goal of the model is the prediction itself. My technique wants to identify what the predictive action sequences are. This is because my models are meant to be used in an adaptive system, which means that prediction on its own is not enough. If I have a model that can predict player type, it does not give any insight on how the game environment should be adapted for this player. Since my model focuses on identifying predictive action sequences, it is easier to translate these insights into in-game adaptions to the environment.

2.2.2.2 Modeling Behavior Outside of Games

Behavior modeling, or user modeling, has received a great deal of attention over the past 30 years. It has been used extensively to quantify and predict user behavior across several
fields. As such, I have chosen to limit my discussion to models that are generated using some of the more popular data-driven techniques.

Decision trees are a simple modeling technique that have the added advantage of being relatively easy to understand. Paliouras et al. (1999) used decision trees to predict user stereotypes, which in this case where the types of news articles that the user was most interested in. In this work, decision trees were used to predict which type of news articles people would most enjoy based on the department in which the user is working, the type of industry in which the user is working, the size of the company they work for, the location of said company, and the location of the market of the company. They received promising results, but claim that their models were not generalizable.

Along a similar line of research, Zhu et al. (2003) used decision trees to model whether certain websites would be useful to a user for completing a specific task. In this case, the task was travel planning. In this work, users would use a web browser that was augmented to allow them to record whether or not the current website they were looking at was relevant to their task. Using these ratings, the decision tree would make predictions based on the users past ratings and characteristics of the web sites that the user visited in the past.

Beck et al. (2003) use decision trees in order to predict whether or not a student will ask for help on certain words in an intelligent tutoring system for reading. To do this, the authors created a set of features about each word in a set of stories that the students were required to read. These were features such as the length of the word and how often it appeared in the story. They also included information about the word's placement in sentences. Student information such as gender, and grade was also considered. They tested this decision tree under two conditions: using data obtained from a group of students to predict whether an unknown student would require help and using a student’s own data to predict whether they would require help. They found that decision trees performed well in both cases, but were especially suited to making predictions using a single student’s data.

Bayesian networks have also frequently been used to model behavior for various purposes. Horvitz et al. (1998) used a Bayesian network to model users interacting with Microsoft Excel. In particular, they predict when a user might possibly require assistance with the program. They also predict what type of assistance is necessary given that assistance was desired.

Miyahara and Pazzani 2000 used a naive Bayesian network to predict user movie reviews based on their rating history. They compared this technique against a correlation-based
approach. The authors found that while the naive Bayesian network was simple, it was still able to outperform the correlation-based approach on this dataset.

Lau and Horvitz (1999) use Bayesian networks to predict various aspects of a user's web search habits. In this work, they use Bayesian networks to infer the probability of a user's next action, the time delay before taking the action, and the user's informational goal.

Gershman et al. (2011) use support vector machines to recommend news articles to users based on their browsing history. The authors divide the user's browsing history into positive and negative examples based on whether or not an article was read or not and use these to train an SVM classifier. This classifier is actually used to determine what the user's current news preferences are. It then uses this classification to find a set of articles that match these preferences and then generates a set of recommendations.

2.2.3 n-Gram Modeling

As I mentioned in Chapter 1, I use an n-gram model of player actions to predict session-level retention (which is discussed in greater detail in Chapter 4). n-gram modeling is a common statistical modeling technique where n past actions are used to predict future actions. These models make use of the Markov assumption that the current action depends only on the n − 1 previous actions that were taken. Although these models are relatively simple, they have been used in many fields. In the following sections, I will discuss how other researchers have utilized n-gram models in games and outside of games.

2.2.3.1 n-Gram Modeling in Games

While n-gram modeling has not seen extensive use in games, it has been used frequently as the basis for action prediction (Millington and Funge, 2009). Action prediction, in the context of a game, is predicting what specific behavior a player (or even an NPC) will perform next. Actions could be things such as attacks in a fighting game or completing a quest. François Laramée (2002) proposed using this technique for use in a fighting game for predicting which attacks a player is likely to perform based on their past history of attacks.

n-grams have also been used to perform goal recognition in interactive narrative environments. Goal recognition is the problem of determining what a player's current goal is based on their action history. This is a difficult problem as actions can contribute to completing many possible goals and it is possible that a player is seeking multiple goals at any given time. Mott et al. (2006) model this problem using an n-gram model of actions. By
doing this, they are able to treat the problem as a classification problem and calculate the probability of an action sequences being associated to a specific goal. The authors tested this method in the interactive narrative environment, Crystal Island. They showed that the $n$-gram models were able to correctly converge (correctly identify the goal based on the action sequence), up to 83.7% of the time.

McQuiggan et al. (2007) used $n$-grams to predict affective state in Crystal Island. Specifically, the authors were interested in predicting whether or not the current player was frustrated. The $n$-gram models studied in this context were able to achieve upwards of 73.6% accuracy in this domain. Additionally, the authors used the insights gained from these $n$-gram models to aid different modeling technique for predicting frustration. They examined the conversion point (the point at which the models consistently and correctly predict the player’s affective state) and used all the data up to that point to train non-sequential classifiers so that they could make early predictions about a player’s affective state. Using this technique, the authors were able to achieve upwards of 88.8% accuracy with an SVM.

Although $n$-gram models have not seen much use in games research, the work that has been done is promising. They have been proven to be effective in making sequential predictions in large game environments and have even been shown to be useful in predicting affective state. These are the main reasons that I have chosen to use $n$-gram models to predict session-level player retention in games.

2.2.3.2 n-Gram Modeling Outside of Games

$n$-gram models have received a great deal of attention in text prediction as they can be used to predict the individual words that come next in a sentence based off of some number of previous words. The $n$-gram model of natural language was introduced by Brown et al. (1992). In this work, the authors outline the idea that word prediction could be done by examining some number of recent words rather than the entire set of past words in a sentence. They also outline the training process by which a training set of sentences is used to calculate the probability that a word will appear given set of words. The original equation they used for this task is:

$$P(w_n|w_1^{n-1}) = \frac{C(w_1^n w_n)}{\sum_w C(w_1^{n-1} w)} \quad (2.1)$$
The equation calculates the probability of a word $w_n$ occurring given the previous $n-1$ words. This probability is calculated by dividing the number of times the string occurs (as is represented by $C(w^n_nw_n)$) by the number of times the prefix $w^{n-1}_i$ occurs (as is represented by $\sum_w C(w^{n-1}_iw)$). This formulation of text modeling is the basis for many of the current techniques used for text prediction.

Expanding on this idea, Pend and Shuurmans (2003) used $n$-gram models to classify sentences. In this case, $n$-gram models are applied to text classification in a similar manner to a naive Bayes model. A text model is created for each possible class and sentences are classified by which class was most likely to generate it. In this work, they examine a three text corpora: a set of 20 works from 10 different Greek authors (totaling 200 works), documents gathered from 20 newsgroups, and a set of news articles published in the Chinese newspaper People’s Daily on a variety of topics. In their experiments, they show that $n$-gram models can achieve upwards of 96% accuracy when predicting a sentence’s class.

Using a similar technique, $n$-gram models have been used in sentiment analysis. Sentiment analysis is commonly used to classify product reviews as either positive or negative based on the content of the review. Chaovalit and Zhou (2005) attempt to use $n$-gram models to classify movie reviews based on their content and whether or not the movie was recommended to other people. The authors achieved an average prediction accuracy of 85.54%, but noted that the recall rates for negative reviews in these experiments were quite poor. The authors claim that the positively skewed dataset that they used may have caused the poor performance classifying negative reviews.

Ye et al. (2009) compare the performance of naive Bayes models, SVMs, and $n$-gram models on a dataset of online reviews to travel destinations. These algorithms were used to perform a sentiment analysis on this set of reviews. They found that, in general, SVMs and $n$-gram models outperformed the naive Bayes model regardless of the amount of training data that was used. With as little as 40 reviews the SVM and the $n$-gram model were able to achieve prediction accuracies of 64.57% and 66.16% respectively. This study shows how powerful $n$-gram models can be on datasets with few examples.

$n$-grams have also been used as the basis for a scheme to conceal packet loss in a voice over IP network. In most networks, packets can be delayed or lost with little to no noticeable effect. This is because the packets can be retransmitted such that all packets reach their destination in order. In voice over IP networks, a lost or delayed packet means that the packet is unusable. To account for this, schemes have been devised which conceal
this packet loss by inserting silence, noise, or simply repeating the last packet received. Lee et al. (2004) use an \textit{n}-gram model to determine which speech utterances should be used to conceal the fact that a packet was lost. They found that using \textit{n}-gram models to predict speech utterances to insert outperforms repetition-based approaches on the same problem.

Shani et al. (2002) use \textit{n}-gram models as the basis for a recommender system. Recommender systems look at a person’s purchase history and make recommendations about products that this person is likely to purchase in the future. This system uses the past \textit{n} purchases to calculate the probability that a user will purchase a given item and uses those probabilities to make recommendations.

These types of models have also been used with some success in goal recognition. Blaylock and Allen (2003) define goal recognition in this context as follows:

\[
G^* = \arg \max P(G|A_{1,j})
\]  

(2.2)

In other words, given a sequence of \textit{j} previous actions \(A_{1,j}\), find the most likely goal \(G\). This is difficult to directly compute as the conditional distributions will grow very large for longer sequences. The authors circumvent this by assuming that an action is dependent only on the goal and the \textit{n} actions preceding it. The authors evaluated this technique using Lesh’s Unix plan corpus (1999), which was gathered from human Unix users that were given a task (or goal) to complete and instructed to solve it using a subset of Unix commands.

These studies show that \textit{n}-gram models have been successfully used for both classification and real-time predictions; however, they have not been used to predict retention either in games or outside of games. The reason that I use \textit{n}-gram models to predict player retention is because of the success that they have seen in classification and action prediction in both games and these other environments.

\subsection{Knowledge Modeling}

A similar concept to player modeling exists in the intelligent tutoring systems (ITS) field. This concept, referred to as knowledge modeling, is concerned with predicting a student’s mastery of a given subject based on their performance while using an automated tutoring system. This is typically done by observing the student as they interact with the ITS and using their performance to determine their individual mastery of certain knowledge concepts. A
knowledge concept is some piece of information or skill that is necessary to answer certain questions posed by the ITS.

For the most part, knowledge modeling requires explicit knowledge of how each KC relates to each question. This means that at least some amount of authoring or expert knowledge is required in order to use many knowledge modeling techniques. For a more detailed discussion of knowledge modeling techniques, please see Appendix A.

2.3 Adaptive Systems

One use for user models or knowledge models is for use in an adaptive system. An adaptive system is any system that is able to alter itself in some way in response to the different types of users that may interact with it. There are several different ways that a system could adapt to different users. In a video game, this could constitute guiding players to content that they are more likely to enjoy. In an ITS, it could be presenting problems in such a way that it enhances learning for different types of people. In the following sections I discuss how adaptive systems have been used in games and in other fields.

2.3.1 Adaptive Systems in Games

In this work, I describe a system for creating adaptive games for increasing session-level retention. Adaptive games are games which can alter themselves in order to influence the behavior or attitude of the player. Below is a summary of work that has been done toward the advancement of adaptive games and examples of adaptive game systems.

The PaSSAGE system (Thue et al., 2008) presents content to a player based on that player’s type. Recall that PaSSAGE has 5 player types that are adjusted as a player moves through the game. When the experience manager presents an action to the player, it looks at the player’s perceived type and then presents the action that most fits that player’s type. So, a player that is predominantly a fighter will be more likely to receive combat related events. The main problem with this approach is that there is no basis in the data for how to update the player’s type as they move through the game world.

Togelius et al. (2007) propose an approach for generating personalized race tracks in a racing game. This system uses a model of player preference and attempts to predict the amount of entertainment a player will experience for a given track and then evolves the
track to optimize entertainment levels. In this system, the tracks are initialized as b-splines and are then mutated by perturbing the position of their control points.

GrailGM (Sullivan et al., 2010) is a system for generating quests or scenarios for a game player to complete. In this system, quests are generated for a user based on that user’s history and the current state of the world. Also, the actions required to finish a quest are not directly stated, which gives the user more freedom in how they want to complete it. While this system does give quests based on the player’s history, it does not make any effort to personalize content based on the player that is currently playing. This is one way in which this work differs from my own.

El-Nasr et al. (2009) dynamically adapt lighting in three-dimensional virtual environments using the adaptive lighting for virtual attention (ALVA) system. This system uses information about possible player goals and the objects necessary to complete these goals in order to control lighting effects to make the objects more visible in a 3-D environment. After comparing ALVA to a static lighting scheme, El-Nasr et al. showed that players playing a first-person shooter using the ALVA system noticed enemies quicker and were less likely to die. This work differs from my own in that it is largely based on hand authored rules about the goals of the user and what is needed to complete them.

Perhaps the most common type of adaptive games exhibit dynamic difficulty adjustment (DDA). These games will determine the expertise level of the current player and then change the difficulty of the game to best suit that player. This differs from the types of adaptions that I make in my system mainly because of intent. These systems are designed to alter challenge directly, whereas my system might only alter challenge indirectly. Through trying to affect other aspects of gameplay (such as play time), there is a chance that challenge could be influenced as a means to that end.

The AI director in Left 4 Dead (Booth, 2009) alters the amount and type of enemies that players will face depending on the perceived amount of intensity that players are experiencing. Intensity is measured by examining statistics such as how many enemies are on the screen and the amount of damage that has been taken by the players. In this system, there is an alternating cycle of rising tension and relaxation. Once the intensity level of the players has risen above a certain level, the number of enemies begins to decrease. Once this relaxation period has continued for a set amount of time, enemies begin to become more frequent until a certain intensity threshold is met and the cycle begins anew.

Another well known example of DDA is the rubber banding technique used in games
such as Mario Kart (Jimenez, 2009). This technique will give noticeable boosts in speed and item quality to players, be they human or computer controlled, that are doing poorly. So, rather than adjust the quality of the AI based on how well or poorly a player is doing, the mechanics of the game are changed to favor those who are not performing well.

Jennings-Teats et al. (2010) alter the environment of Infinite Mario by adding pits and different platform levels in order to dynamically adjust the difficulty. In this system, a statistical model of player skill is used to determine the placement of enemies and the placement of obstacles to ensure that the player is sufficiently challenged.

Andrade et al. (2003) proposed the use of reinforcement learning as a way to teach agents to dynamically adjust their skill in a fighting game. In their system, a reinforcement learning agent takes actions (punches and kicks) in such a way that the fight ends with both combatants having a small difference in life totals, indicating that the agent was a challenging opponent, but not so much as to cause frustration in the player.

Hunicke and and Chapman (2004) use a probabilistic model of player success/failure and parameter manipulation to adjust the difficulty of a first-person shooter. Their model, which will probabilistically determine if a player is flailing (moving towards a state where the player is ill-equipped to handle the challenges that lie ahead) and then make alterations to the game world to make these challenges doable. These alterations include making enemies weaker, making the player's weapons more effective, and making more health packs available to the player.

Spronck et al. (2004) use dynamic scripting (Spronck et al., 2003) to adjust difficulty by shifting the way that rules are chosen in their system. Their system has three ways to adjust difficulty: high-fitness penalizing, weight clipping, and top culling. High-fitness penalizing rewards the agent for choosing actions that have a moderate fitness value rather than those that have high fitness values. Weight clipping puts a limit on the probability that certain rules will be selected which increases the chance that the agent will choose suboptimal moves. Top culling makes it impossible to choose rules that perform too well, forcing the agent to take suboptimal actions.

2.3.2 Adaptive Systems Outside of Games

Adaptive systems are also quite prevalent in fields outside of games. Probably the field that is most related to my research is that of drama management. According to Roberts et al. (2008), a drama manager is an omniscient coordinator that directs objects and characters
in the game world to influence the plot progression. In other words, a drama manager seeks to make alterations to a story world and characters to bring about some change or to ensure that a certain plot comes to fruition. The main difference between my work and that of a drama manager is that a drama manager seeks to influence only the plot of a story whereas I seek to influence player behavior for a variety of reasons such as increasing play time or improving player experience.

Despite this difference, it is still important to be aware of the advances made in drama management. Perhaps the most famous example of a drama management system is the one used for the interactive narrative, Façade (Mateas and Stern, 2003). Façade uses a beat-based drama manager that tries to fit a global plot arc for a particular story. Façade's drama manager determines where it is in the global plot arc, and will then apply beats until the necessary amount of change has occurred for it to continue to fit the desired arc.

El-Nasr (2007) uses beats in conjunction with a model of the current user in order to adapt a story world to maximize engagement, dramatic content, and the quality of interactive narratives. The beats used in this system were created to encode various techniques popular in performance arts. In this work, narrative beats were selected based on the user model and which beats had been executed in the past.

The interactive drama architecture (IDA) created by Magerko and Laird (2003; 2004) uses drama management to ensure that player actions do not threaten the completion of a set of author defined story goals. So, if the player performs an action that threatens the completion of a goal later in the story, the drama manager must intervene in some way in order to either prevent the player from performing that action, or enable the completion of said story goal through another means. The IDA uses SOAR agents (Laird et al., 1987) that allow the story director to issue high level instructions and allow the agent to carry them out autonomously.

Mott and Lester developed U-Director (2006), a system that is designed to handle the uncertainty introduced by player autonomy. U-Director is intended to maximize narrative rationality, which is reasoning about narrative objectives, the story world state, and user state in an uncertain world in order to maximize narrative utility. U-Director does this by using a structure called a dynamic decision network (DDN). A DDN is a generalization of a Bayesian network that also includes nodes for utility and choice. The structure of the network defines the effect that certain actions will have on the narrative state and examines actions in 3 different time slices: the current game state, the game state after the director
has taken an action, the game state after the player’s reaction to the director’s action. In order to make a decision, the director updates the network with the current information and performs the action that will result in the highest narrative utility in the third time-slice. This process repeats whenever a decision needs to be made.

The Mimesis system (Young et al., 2001) is a drama management system that uses planning to determine the possible stories that will satisfy a set of author defined goals. This system accounts for interaction and, therefore, adaption through the use of alternate plans or through intervention. In this system, the user takes actions until the user attempts to take an action that will threaten one of Mimesis’s current plans. Mimesis handles this by either selecting an alternate plan that will still accomplish the story goals set out by the author. If no such plan is available, then Mimesis will cause the user’s action to fail. An example of an intervention is a gun jamming to prevent the death of an NPC important to the story.

Barber and Kudenko (2007a; 2007b) created a drama management system that uses dilemmas to create interactive stories. In this system, the authors created 5 dilemmas, or choices that the user is forced to make that will have noticeable effects on the player’s relationship with the involved NPCs, that the drama manager uses to adapt the story. In order to create a story, the drama manager selects a dilemma based on how appropriate it is to the current situation and how frequent a dilemma has occurred in the recent past. The appropriateness of a dilemma is determined by examining how the player has reacted to similar dilemmas in the past. This is then used to determine how difficult a certain dilemma will be for the player. The drama manager is then more likely to select difficult dilemmas for the player to face.

The drama management systems that are most like my own are targeted trajectory Markov decision processes (TTD-MDPs) (Roberts et al., 2006) that are built on the declarative optimization-based drama management formalism (Nelson et al., 2006). TTD-MDPs were created to solve the problem of replayability in an adaptive story world. TTD-MDPs target a distribution of stories, or trajectories through the space of plot events, and attempt to form a policy such that this distribution of stories, as opposed to the best story, will occur.
2.4 Background on Psychometrics

In order to help motivate my choice of qualitative metrics as well as position the work presented in this dissertation, the following section will provide information on intrinsic motivation and engagement.

2.4.1 Intrinsic Motivation

For the purposes of this dissertation, I use the definition of intrinsic motivation put forth by Deci et al. in (2000). According to Deci et al., intrinsic motivation is defined as the doing of an activity for its inherent satisfactions rather than for some separable consequence. In other words, an intrinsically motivated individual will perform an action based on enjoyment or challenge of performing the activity rather than based on external pressures or rewards.

In (Deci and Ryan, 2000), Deci and Ryan review three psychological aspects of intrinsic motivation: autonomy, competence, and relatedness. Autonomy is the idea that you are performing an activity because you have chosen to. Factors such as external rewards and pressures inhibit this feeling of autonomy, which makes designing effective experience managers quite difficult. At a glance, it seems that the goal of the experience manager is in conflict with this aspect of intrinsic motivation; the experience manager wants to control player behavior, and yet the player feeling controlled leads to a low sense of autonomy. What this actually means is that the experience manager is limited in the actions it can take to influence player behavior. If the experience manager makes only subtle changes to the game environment, then the player will be unaware that she is being manipulated and maintain a sense of autonomy.

The need for competence is the need to have an effect on the environment and to be effective in one's interactions with the environment (Deci and Ryan, 1985). In other words, competence in a game environment is closely related to the difficulty of the game. Players experiencing high competence will be presented with tasks that are challenging, but not so challenging that the task becomes insurmountable. It has also been proposed that competence will only increase intrinsic motivation if the feelings of competence do not come at the expense of player autonomy. So, if the player feels manipulated into a competent experience, it is likely that they not feel an increased sense of intrinsic motivation.

The final aspect of intrinsic motivation, relatedness, is also considered to play a more distal role in intrinsic motivation. Relatedness refers to being accepted by one’s peers and
being encouraged by them when performing an activity. It has been noted that very often people will engage in intrinsically motivating activities (such as hiking) which occur in isolation, which leads many people to believe that relatedness is less centrally related to intrinsic motivation than autonomy or competence. It is for this reason, along with the inherent difficulties in addressing relatedness in a single-player game environment, that I choose not to consider relatedness when determining how my adaptations will influence intrinsic motivation.

I chose this measure of player experience because I believe that the alterations will increase the player’s perceived competence while still maintaining player autonomy. The adaptations that I will make to the game, while not designed specifically to effect challenge, will likely make the game seem easier. For example, guiding a player towards content they are more likely to enjoy could raise competence as this player may enjoy content that she is more familiar with or more skilled at overcoming. I expect that this increase in perceived competence will manifest itself as an increase in intrinsic motivation.

2.4.2 Engagement

The concept of engagement is a broad term used to describe player involvement. Over the years, many measures have been used to study engagement (Appleton et al., 2006; O’Brien and Toms, 2008). In this paper, I focus on the use of four measures of engagement in particular: immersion, presence, flow, and absorption. Immersion, considered to be a determinant of shallow engagement, measures engagement in games while the player maintains an awareness of their surroundings. So, a player who is immersed in a game will be aware of their surroundings in the game, but still be aware of the surrounding environment and response stimuli located in the real world.

While the consensus on the definition of presence is still being reached, most definitions include being in a normal state of consciousness while having the experience of being inside of a virtual environment. Tamborini and Skalski (2006) define presence as a sense of “being there” in reference to a virtual environment. In general, presence is considered to be a more intense form of engagement than immersion, but is still considered shallow overall (Lombard and Ditton, 1997).

Flow is a concept that is not new to games research (Sweetser and Wyeth, 2005). Flow is defined as an altered state of consciousness that occurs when performing an intrinsically motivating activity that is sufficiently challenging (Moneta, 1996). It has been shown that
there are several contributing factors to achieving a flow state. In addition to the challenge level of the activity, having clear goals and immediate reward feedback can help players enter a flow state. A common characteristic of being in a state of flow is the distortion of time. In other words, time seems to either slow down or speed up while performing an activity in a state of flow. Since being in a flow state is considered being in an altered state of consciousness, it is accepted that players who experience flow are more engaged in a game than those who are simply immersed or have a feeling of presence.

Finally, absorption is considered to be one of the most extreme forms of engagement. It has been described as “total engagement in the present experience” (Irwin, 1999). Absorption is considered to be an altered state of consciousness in which there is a separation of thoughts, feelings, and experiences (Glickson and Avnon, 1997). In other words, during absorption the thoughts and feelings of the player become separate as the player merges with the virtual environment that they are playing in.

It has been stated that these measures of engagement can be seen as a hierarchy denoting an increasing sense of engagement in games (Brockmyer et al., 2009). So, immersion represents the shallowest form of engagement while absorption represents the strongest sense of engagement.
CHAPTER 3

ACTIONABLE AND VANITY ANALYTICS

3.1 Introduction

In Chapter 1 I introduced two types of analytics: actionable analytics and vanity analytics. These terms will be used frequently in the remainder of the document to refer to the types of analytics I use for both modeling and for making dynamic game adaptations. This chapter contains a detailed discussion on what exactly vanity analytics and actionable analytics are and what they might look like in a game environment. I will also discuss the two game environments that I will use as case studies, Scrabblesque and Sidequest: The Game

3.2 Actionable and Vanity Analytics

The terms actionable analytics and vanity analytics are used frequently in business intelligence (Ries, 2011) to describe the different types of analytics gathered. Actionable analytics are those that can be used to perform prescriptive analyses. A prescriptive analysis is an analysis that gives recommendations on specific actions that should be taken to bring
about specific outcomes. In other words, actionable analytics are those that give people a clear idea of what to do with them. Often times, this means that the people making these decisions have direct control over the values of these analytics. Returning to the political metaphor introduced in Chapter 1, business analysts use actionable analytics much like politicians use money. Recall that in the metaphor the only way that a politician can affect the polls is to distribute money to buy advertising time in an attempt to affect poll voters. In other words, the only things that politicians can directly affect is the amount of money that is being distributed and where that money goes.

Vanity analytics exist at the other end of the spectrum. In business intelligence, vanity analytics are those that serve to describe the current state of a product. Typically, they do not give any indication of what led to their value or what can be done to affect them. They earned the name vanity analytics because they are typically used to show a product’s effectiveness or general success. The number of product downloads or number of unique users using a product are common vanity analytics found in business intelligence. In the political metaphor from Chapter 1, the polls can be thought about as vanity analytics. They are a good indicator about the state of the election, but it is unclear how to directly affect these values. At best, politicians can indirectly influence these values by intelligently distributing their money. Moving back into the realm of business analytics, similar behavior is observed by considering a popular vanity analytic, number of product downloads. By looking at the number of times a product was downloaded it is possible to get an idea how many people are using it, but it is impossible to say what is causing this or how to affect this value. In this example, these analytics are purely descriptive.

### 3.3 Analytics in Games

The terms actionable analytics and vanity analytics are commonplace in business analytics, but what do they look like in a game environment? In the following sections, I will discuss what these analytics might look like in a game environment and give some descriptive examples of each. For the purposes of this dissertation, I am choosing to focus on actionable and vanity analytics that specifically describe the in-game environment and in-game behaviors.
3.3.1 Actionable Analytics in Games

Recall that actionable analytics are those that can be used to determine a clear course of action. The reason that these analytics provide such courses of action is typically because they are directly affectable. In other words, it is possible to control or manipulate the values of these analytics. Thus, these types of analytics in games constitute aspects of the game that designers can directly control or manipulate. These analytics can differ from game to game based on how possible changes would be implemented. For example, consider a first-person shooter game in which the authors will implement changes off-line (such as through online patches). This type of game could contain analytics involving map construction (such as the placement of rooms or walls), power-up placement, or the difficulty of enemy AI. If designers want to implement dynamic changes, however, then they can no longer change map construction. This is because introducing very visible changes to fundamental aspects of the environment that are assumed to be static can introduce dissonance between the player’s perceived model of the world dynamics and what the actual dynamics are. This dissonance could have a negative impact on the player’s experience (Young, 2002). As such, actionable analytics for dynamic game adaptations tend to be more subtle than those used for off-line game adaptations.

While these analytics are, typically, easy to directly affect, they are difficult to directly use in either a descriptive or predictive modeling effort. This is because these features exist without reference of how the player interacts with them. In other words, they often do not contain enough inherent information to be useful in modeling. These types of analytics also often have several possible values, making them vulnerable to the curse of dimensionality. Consider an analytic describing power-up placements on a map. There are many possible placements of health packs on a map. In order to use this specific analytic in a predictive modeling effort, it would require an exponential number of observations just to model the possible space of placement configurations. This means that it would take a great deal of effort just to gather the requisite number of observations to even model the space.

3.3.2 Vanity Analytics in Games

Contrary to actionable analytics, vanity analytics are typically very easy to observe and model. In games, these analytics usually deal with the state of the player or player behavior in some way. Consider the example used in the previous section about a first-person shooter
game. Examples of possible vanity analytics include the number of enemies killed, the number of power-ups picked up, or the number of times the player fired their weapon. The key difference between vanity analytics and actionable analytics is that it is difficult to directly affect vanity analytics. Consider each of the examples that I listed above. As a designer, there is no way to ensure that the player will pick up more/less health packs or kill more/less enemies. At best, vanity analytics can be indirectly influenced through the manipulation of actionable features.

For another example, consider Figure 3.1. This figure shows a heatmap of player death location and frequency in a level of Valve's game *Half-Life 2*. In this figure, green, yellow, and red areas represent places where players died. The color refers to the frequency that players died in those areas with green representing the least amount of deaths and red representing the most amount of deaths. These analytics provide a great deal of descriptive insight into player behavior; however, they offer very little in terms of prescriptive insights since there is no way to directly affect these analytics.

### 3.4 The Gap Between Analytics

The previous sections lend credence to the following idea: there exists a gap between the descriptive/predictive power of vanity analytics and the prescriptive power of actionable analytics that needs to be bridged if the insights gained from vanity analytics are to be incorporated into a game environment. The descriptive and predictive power of vanity analytics do not give designers enough direction to translate their insights directly into game adaptations, and actionable analytics are often not powerful enough on their own to generate the insights necessary to put their prescriptive power to use.

Until a way is found to bridge this gap, it will be difficult to use models created using vanity analytics to generate game adaptations. For example, consider the problem that I am addressing in this dissertation: session-level retention. Let's assume that I am able to identify player behaviors that are predictive of them ending a game session before it should normally end. As it stands, there is no real way to use this information about session-level retention to generate game adaptations. I can not directly control a player's behavior while still maintaining the idea that the player's actions are meaningful within the context of the game, and this model is of little use as a result.

---

1[^1]: [http://orange.half-life2.com/hl2.html](http://orange.half-life2.com/hl2.html)
Figure 3.1 A heatmap showing player deaths in a level of Half-Life 2
In order to bridge the gap, I use the insights gained from vanity analytic models to determine how certain actionable analytics should be altered. For a better understanding of this, let's return once again to the political example that I have been using. In this case, the strategy that I use is equivalent to using the results of the polls to inform how to distribute campaign funds. Using this strategy, there is still no guarantee that the money will be enough to actually influence the polls; however, it still allows politicians to leverage the predictive power of the polls to find intelligent ways to spend their money. Relating this back to my dissertation, I use the insights provided by predictive models constructed using vanity analytics to intelligently determine how to manipulate actionable analytics in order to influence player behavior in a desirable way.

3.5 Game Environments

In this dissertation, I will be presenting two case studies in which I influence player behavior in order to improve session-level retention. These case studies involve two game environments that I created: Scrabblesque and Sidequest: The Game (SQ:TG). In Chapter 1, I gave a high level overview of each of these environments. In this section, I want to further motivate the use of these environments and go into detail about the vanity and actionable analytics that I measure in each one.

3.5.1 Motivation

The game environments that I created are, on the surface, very different. Scrabblesque is an implementation of the board game Scrabble in which you play against a single computer controlled AI, whereas SQ:TG is a quest-based adventure game in where your goal is to complete a certain number of quests. While there are many differences between the environments, what is important is that they both fall into the realm of casual games. Casual games are games that do not take a very long time to complete and have relatively simple mechanics that most people can easily learn. They are also identified by a lack of commitment (Juul, 2012). In other words, it is easy to start playing and there is not a very large penalty for ending play. Both of the games that I have created fall into this category. Both offer limited ways to interact with the environment and the controls are relatively simple to pick up and master. They can also be completed in less than 30 minutes, meaning that there is minimal time investment in order to complete each of them.
What is interesting is that these game environments represent two different game genres while still both being casual games. This is done to examine how my strategy for bridging the gap between vanity and actionable analytics would generalize to other types of game environments. While this does not give definitive proof that this strategy will work in every type of game, it will provide evidence that it could be useful in other genres.

3.5.2 Vanity Analytics and Actionable Analytics in Scrabblesque

In Scrabblesque (see Figure 3.2), players take turns playing letter tiles onto a static game board in order to form words. Based on the letters in the words that are formed and where they are placed on the game board, each player will earn points. In a game like this, there are several vanity analytics that I can use for modeling. Examples of possible vanity analytics include frequency of playing certain letters, length of time playing the game, and player score. For the analysis I perform in Chapter 4 I use the following vanity analytics:

- Score difference: The difference between the player's and the computer's score
- Turn Length: The amount of time that has passed from the beginning of a turn to the end of that turn (characterized by the final word that is accepted by the game)

Figure 3.2 A screenshot of Scrabblesque. The game board has been outlined in red and the player's rack of letter tiles has been outlined in blue.
• Word Length: The length of the last word played

• Word Score: The point value of the last word played

• Word Submitted: The length of time in between word submissions (because players can submit as many words as they want until one is accepted by the game, I looked at submissions separately from acceptances)

I chose these vanity analytics because they give a comprehensive representation of player behavior in Scrabblesque. Intuitively, they can also be tied to session-level player retention. For example, if the time between words submitted decreases it could be a sign that the player is losing interest and not thinking about their future actions. This could, feasibly, lead to the player ending their game session.

It is important to note that, as with vanity features in general, there is no way for these features to be directly manipulated either dynamically or off-line. In order to use any models created using these analytics, I need to identify what aspects of Scrabblesque are under my control.

It turns out that there are very few things that the designer can directly control in the game. In Scrabblesque, there are 3 entities that the designer can feasibly control: The game board, the opponent, and the tiles that the player receives. Since I am concerned with dynamic game adaption, I can disregard the game board. This is because the game board remains static during gameplay (except for the addition of letter tiles through normal gameplay). The other two entities, however, are reasonable actionable analytics. I record different sets of actionable analytics for each of these entities. The analytics describing the words that the computer plays are:

• Number of Candidate Tiles: The number of eligible tiles on the game board that the player can use to form words

• Consonant/Vowel Distribution: The distribution of consonants and vowels amongst candidate tiles

• Average Tile Value: The average value (in terms of game score) of candidate tiles

• Proximity to Bonus Squares: The number of bonus squares that the player can reach in a single turn
Since players can only make words by incorporating letters in the words that the computer plays, these analytics measure the properties of the tiles that players can play off of. Contrary to the set of vanity analytics I examine, each of these can be controlled by altering the word that the computer controlled AI plays. If the computer chooses a different word to play, the values of these analytics will change.

In addition to these analytics, I also measure the following set of actionable analytics describing the state of the players rack:

- Consonant/Vowel Distribution: The distribution of consonants and vowels present on the player’s rack
- Average Tile Value: The average value of the tiles in the player’s rack
- Number of Repeated Tiles: The number of tiles that are repeated in the player’s rack

As with the analytics describing the words that the computer plays, these analytics are controlled by the tiles that the computer gives the player. Once the player successfully places a word on the game board, their rack of letter tiles is refilled. Traditionally, tiles are given to the player at random from a set of all remaining tiles. Based on which tiles are given to the player, the values of these analytics will change, meaning that the AI has a way to easily manipulate these values by controlling which tiles the player receives.

### 3.5.3 Vanity Analytics and Actionable Analytics in Sidequest: The Game

Sidequest: The Game (see Figure 3.3) is a significant increase in the amount of interaction afforded to the player when compared to Scrabblesque. As such, there are more possible vanity analytics that I can consider when modeling session-level retention. Examples of possible vanity analytics include the number of enemies killed and the number of non-player characters (NPCs) that the player interacted with. For my modeling efforts in Chapter 4, I consider the following vanity analytics:

- Action Type: The action that was performed. Can either be accepting a quest, rejecting a quest, completing a quest, or abandoning a quest.
- Action Times: The amount of time that passed between the last action and the current action.
Figure 3.3 Quest-givers in *Sidequest: The Game*. For ease of understanding, characters that have quests to offer the player are circled in yellow. The player is circled in blue.
The reason that I chose only these two analytics is because interacting with quests is central to SQ:TG. As such, I want to examine analytics that can describe how players interact with quests. It is also feasible that these features are intertwined with session level retention. For example, a player rejecting too many quests in a row could be a sign that they are not finding quests that they would enjoy, which could lead to them quitting the game. Similarly, if I see a person taking a long time to complete certain quests, then it is possible that they are losing interest in the game and will quit soon.

In addition to more types of vanity analytics, SQ:TG also affords me more possible actionable analytics as well. Things such as the position of quest-givers in the world, the frequency and difficulty of enemy NPCs, and the location of quest goals in the game world are all actionable analytics that exist in SQ:TG. For this dissertation, I choose only to use the proximity of a quest to the player as the actionable analytics that I will control. To fully understand this analytic, please refer to Figure 3.3. This figure shows a screenshot from SQ:TG depicting the player (who is circled in blue) in a room with 5 quest-giving NPCs (who are circled in yellow). When I refer to the proximity between a quest and the player, I mean the distance from the player’s character to the NPC that gives out that quest. In SQ:TG, it is not possible move an NPC, but what I can control is which quests each quest-giving NPC makes available to the player. In this way, I can directly affect the relative proximity of each quest to the player relative to every other quest.

3.6 Conclusion

This chapter has gone into greater detail about vanity analytics and actionable analytics and shown what these analytics look like in a game environment. More importantly, I have shown that a gap exists between these two types of analytics with respect to game adaption. Actionable analytics provide a clear path for change. They describe aspects of a system (in my case, the system is a game) that designers have direct control over; however, they often lack the predictive power to generate useful insights since they do not give any context as to how the user is interacting with the system or how the user is behaving. They are also quite vulnerable to the curse of dimensionality, which further limits their use in predictive modeling. Vanity analytics, however, have a great deal of predictive power and are used quite frequently in modeling efforts. The problem with them is that they are, oftentimes, outside of our control. In order to dynamically adapt game environments, techniques need
to be developed that bridge this gap between vanity and actionable analytics.
4.1 Introduction

In the previous chapter, I expanded on the terms *vanity* and *actionable* analytics and gave some examples of what these look like in game environments. Recall that vanity analytics often hold great predictive power and can be used to create powerful models of human behavior. Also recall that the work in this dissertation is aimed at improving session-level player retention. In this chapter I will introduce a technique for modeling session-level retention and detail how I use the vanity analytics described in the previous chapter to create data-driven models in both *Scrabblesque* and *Sidequest: The Game.*
4.2 Raw Vanity Analytics

Before game adaption can occur, one must have a model of player behavior. In my case, specifically, I required a model describing player behaviors and whether or not they are associated with players quitting the game before it would naturally conclude. As I said in the previous chapter, vanity analytics typically have enough predictive power to perform such a task. As such, I chose to examine a set of vanity analytics and use them to create a model of player behavior that could predict whether or not a player was likely to end the game before its natural conclusion.

My initial modeling hypothesis was that I could treat this as a supervised learning problem (Russell et al., 1995). Supervised learning is a type of machine learning problem in which a computational model is learned from a set of labeled data. Specifically, I viewed this as a classification problem. By examining traces of player behavior (as described by a set of vanity analytics), this model should be able to predict a player's class in real time as the player interacts with the game world. In this model, a player's class can be many different things. Examples of possible player classes include player type, whether a player will exhibit a certain behavior or not, and demographic information. For the purposes of my work, I use player class to represent whether a player will end a game session prematurely. The reason that I chose to treat this problem as a classification problem is because there are several examples of previous researchers that have treated this problem similarly both in game research and outside of game research.

In the following sections, I describe the results of a set of experiments that I performed in order to identify player class in Scrabblesque. In these experiments, I use the set of vanity analytics discussed in conjunction with various classification techniques to attempt to predict whether or not a player will end their game session prior to its natural conclusion.

4.2.1 Modeling Session-Level Retention in Scrabblesque

My initial hypothesis was that I could use vanity analytics as they were gathered to predict player class by treating it like a typical machine learning classification problem. In other words, I would look at each analytic's raw value and then use this as the input to a classification algorithm.
4.2.1.1 Data Collection and Experimental Methodology

The purpose of this experiment was to see if vanity analytics in *Scrabblesque* could be combined to predict whether or not a player would end the game early using typical machine learning techniques.

In order to create this model, I use the set of vanity analytics described in Chapter 3. As a reminder, the list of vanity analytics that I will use create this model are:

- Score difference
- Turn Length
- Word Length
- Word Score
- Word Submitted

As mentioned earlier, these analytics were chosen because they provide a comprehensive representation of player behavior and are, at least intuitively speaking, related to session-level retention in some way. Another interesting aspect of these analytics is that many of them are common to other casual games as well. Score difference and turn length, for example, are both very generic and can be measured in other turn-based casual games. Word length, word score, and words submitted are specific to Scrabble; however, they do have more generic analogs in other casual games, like action complexity, previous round score increase/decrease, and action interval respectively. This means that there is a chance that the models built using this could generalize to other games without having to retrain the models. While this could be the case, it is beyond the scope of this dissertation and is left for future work.

To perform these experiments, I performed a data collection over the course of one month. At the end of the data collection, I had gathered 195 game traces. I recruited participants using snowball sampling from mailing lists at North Carolina State University, social networking sites, and popular technology forums on the Internet. In total, 94 different users played, which indicates that users played multiple games. A summary of my dataset can be found in Table 4.1.

In addition, each game was tagged as either finished or unfinished based on whether or not the player ended the session prematurely. If the game was allowed to continue until
Table 4.1 Summary of Scrabblesque dataset. Games played shows the number of games played in each category. Game length reports the average game length and standard deviation in game turns. Players shows the number of unique players in each category.

<table>
<thead>
<tr>
<th></th>
<th>Games Played</th>
<th>Game Length</th>
<th>Players</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finished</td>
<td>148</td>
<td>10.1 ± 2.5</td>
<td>56</td>
</tr>
<tr>
<td>Unfinished</td>
<td>47</td>
<td>4.4 ± 3.3</td>
<td>49</td>
</tr>
</tbody>
</table>

either the computer player or the human player had won, then the game was marked as finished. If the game ended before a winner was declared, then the game was marked as unfinished. Once this was completed, 148 games had been tagged as finished while 47 games had been identified as unfinished.

Since the purpose of my player model is to predict session level retention in real time, I treat this data as a time-series. In order to do this, the values of each of the analytics discussed earlier are calculated for each turn and used to create a time-varying data set. As a result of this process, I have training sets representing 21 turns worth of game play. Thus, after each of a player’s turns, I can attempt to classify each player based on their actions taken up until that turn. Although I have data for 21 turns of gameplay, it does not guarantee that there are a similar amount of actions taken on each turn. This means that training sets for later turns may end up being too data-sparse for modeling.

For these experiments, I chose to use these vanity analytics as inputs into three classification techniques and evaluated their resulting prediction accuracies after each turn. Specifically, I used a Bayesian Network (Friedman et al., 1997), Multilayer Perceptron (Orbach, 1962), and C4.5 Decision Tree (Quinlan, 1993). Since the different biases inherent in these algorithms make their suitability for individual data sets different, I felt they would be reasonably representative of existing classification methods. I assume that any measures made to keep the player playing will not have any adverse effect on players that were not going to quit prematurely; however, if I misclassify an unfinished game as finished, then no measures can be taken to retain that player, which is what I want to avoid. Also, this particular dataset is unbalanced towards finished games, meaning that simply looking at predictive accuracy can be misleading since high accuracy can be achieved on this dataset by simply classifying everything as a finished game. Thus I am mostly interested in my ability to accurately predict an unfinished game. To test each of these methods I used 10-fold cross-validation and recorded the classification accuracy.
Before the experiments were run, all input values were discretized into 3 bins: low, medium, and high. These bins were created by using the following equation:

\[
D(f_i(p, j)) = \begin{cases} 
  \text{low} & \text{if } f_i(p, j) < B_{i,j} \\
  \text{med} & \text{if } f_i(p, j) < 2B_{i,j} \text{ and } f_i(p, j) \geq B_{i,j} \\
  \text{high} & \text{if } f_i(p, j) \geq 2B_{i,j}
\end{cases}
\] (4.1)

In the above equation, \(D(f_i(p, j))\) is the discretization function given the value \(f_i(p, j)\) of feature \(i\) on turn \(j\) for player \(p\) and \(B_i\) is the size of each bin for feature \(i\). \(B_{i,j}\) is calculated by the following equation:

\[
B_{i,j} = \frac{\max_{t=0,\ldots,i}(f_t(p, j)) - \min_{t=0,\ldots,i}(f_t(p, j))}{3}
\] (4.2)

As you can see in the above equation, bin size is calculated by considering all values up to the current turn to determine the max and min values. I consider values that are greater than two standard deviations away from the average value of a feature as outliers and do not include them when determining the bin size. I do this to prevent outliers from skewing the size of each bin. These values are still used in the model learning and class prediction processes. As you can see in Equation 4.1, the discretization function \(D\) transforms every continuous value for every feature into either low, medium, or high. This technique for calculating bins will be relatively robust to outliers except in degenerate cases such as bimodal distributions of values. I empirically verified that my data did not fall into these degenerate cases.

### 4.2.1.2 Results and Discussion

The results of the cross validation over the course of 6 turns can be found in Table 4.2. For reported prediction accuracy, I am only considering how well the algorithms predict if a player will quit the game early and as such, I am only reporting each classifier’s performance on those games. The reason that I only show results for the first 6 turns is because the prediction accuracy for unfinished games does not improve after that turn. For evaluation, I compare against the baseline of 24.1% prediction accuracy. This was chosen as my baseline because the \textit{a priori} probability of any individual game resulting in early termination is 0.241 in my training data. If my models can produce a predictive accuracy of greater than 24.1%, then I am making predictions that are better than a random guess. As can be seen in
Table 4.2 Prediction accuracies for predicting session-level retention on each turn in Scrabblesque. After the turn 6 the prediction accuracy never increases. Values that are greater than the prediction threshold are bolded.

<table>
<thead>
<tr>
<th></th>
<th>Turn 1</th>
<th>Turn 2</th>
<th>Turn 3</th>
<th>Turn 4</th>
<th>Turn 5</th>
<th>Turn 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bayes Net</td>
<td>0.32</td>
<td>0.23</td>
<td>0.08</td>
<td>0.10</td>
<td>0.05</td>
<td>0.00</td>
</tr>
<tr>
<td>Perceptron</td>
<td>0.17</td>
<td>0.25</td>
<td>0.22</td>
<td>0.13</td>
<td>0.10</td>
<td>0.15</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Table 4.2, these techniques result in prediction accuracy well below the baseline in all but two cases.

These results show that this problem cannot be solved using traditional classification techniques. It also calls into question how informative the base vanity analytic values are in predicting whether a player will end the game early or not and, therefore, motivates the use of different types of vanity analytics for modeling session-level retention.

### 4.3 n-Gram Modeling Using Deviation-Based Analytics

In response to the poor performance of the aforementioned raw analytic values in the previously described experiments, I hypothesize that perhaps using information about how each analytic value differed from that of a normal player would be more informative than using just the base values of the analytics. For example, a player whose turns suddenly start to last longer than most normal players’ turns could be more predictive of player retention than just knowing how long each turn lasted. In order to achieve this, I need to find a way to encode this additional information into each analytic value.

To capture this extra information, I chose to have each analytic represent how much it differed from its observed average value. Using the example presented in the previous paragraph, the feature for turn length would refer to how close to the average length of a turn the current player is rather than simply the length of their turn. This process is better known as a symbolic aggregate approximation (SAX) transformation (Lin et al., 2003).

In addition, it is possible that standard classification algorithms were not powerful enough to predict session-level retention in this environment. I hypothesize that this is because classification algorithms, as I was using them, ignore any relationships that may exist in how each analytic value changes over time. As a result, I have chosen to model
this problem using \( n \)-grams to capture any temporal relationships that exist for this set of analytics. These models make the assumption that the current state of the world depends only on a small subset of the state history. Specifically, the current state depends only on the previous \( n-1 \) states. Using this information, I determine which sequences of states (or analytic values in my case) are predictive of players quitting the game early.

In the next sections, I will describe how I transform baseline analytics into deviation-based analytics using the SAX transformation in greater detail as well as how to use these analytics to create an \( n \)-gram model for predicting session-level retention.

### 4.3.1 SAX Transformation Methodology

In order to use the vanity analytics that I have chosen in this model, I must first perform a SAX transformation on each analytic value. For each of the input analytics, I use the following formula to perform the SAX transformation:

\[
a'_{i}(p, j) = \left| a_{i}(p, j) - \frac{\sum_{p'} a_{i}(p', j)}{k} \right|. \tag{4.3}
\]

In the above equation, \( a_{i}(p, j) \) is the value of analytic \( i \) on turn \( j \) for player \( p \) and \( k \) is the number of players in the training set. \( a'_{i}(p, j) \) is the absolute value of the difference between the player \( p \)'s value for feature \( i \) on turn \( j \) and the average value for analytic \( i \) on turn \( j \) across all players. I then discretized \( f'_{i}(p, j) \) into equal-sized bins using Equation 4.2.

This process produces separate training sets each corresponding to a different analytic. To avoid issues with data sparsity, I choose to consider each analytic in isolation. This means that every input feature will produce a model specific for that feature. For example, using analytics describing “trap-based deaths” and analytics describing “enemy-based deaths” as inputs would produce one model for “trap-based deaths” and a different model for “enemy-based deaths”.

It may seem that this method for producing player models will become very difficult to manage as the number of input features increases. The trade-off present here is one of the difficulty managing multiple models versus data sparsity. For the purposes of my research, I determined that data sparsity was a much larger issue than that of managing multiple models. Due to the curse of dimensionality, a relatively low number of features can lead to an explosion in the amount of observations needed to sufficiently describe the search
space. For these reasons, I have chosen to examine each feature in isolation.

### 4.3.2 n-Gram Modeling

My revised hypothesis is that these transformed expectation-deviation analytics can be used as predictors for determining player class. The next thing I need to do is find a way to incorporate temporal information into the model. To account for this, I decided to calculate the probability that the player belongs to a given class using the following equation:

\[
P(c|i, s) = \frac{P(i|c)P(s|c)P(c)}{P(i, s)}
\]  

(4.4)

In the above equation, \(c\) is the player class label, \(i\) is the turn number, and \(s\) is the complete sequence of previous analytic values. This technique becomes more susceptible to the curse of dimensionality as the length of an action sequence increases. In order to use this model effectively, I would need observations for each possible sequence of analytic values, which quickly becomes infeasible as the length of that sequence grows. In fact, the problem would require exponentially large training sets to effectively model the space of possible analytic value sequences. To address this, I make a Markov assumption that the current value for a specific analytic depends only on a small subset of previous analytic values. This means that a player’s entire analytic value history can be described using their most recent \(n\) actions. This means that Equation 4.4 can be rewritten as

\[
P(c|i, s_n) = \frac{P(i|c)P(s_n|c)P(c)}{P(i, s_n)}
\]  

(4.5)

In this equation, the player’s analytic value sequence \(s\) is replaced with only the last \(n\) analytic values \(s_n\). It is often difficult to directly calculate \(P(c|i, s_n)\); however, it is much easier to calculate the other probabilities in the equation which are:

- \(P(i|c)\): The probability of it being turn \(i\) given the class label \(c\). This is calculated by finding the number of games that lasted at least until turn \(i\) in class \(c\) compared to the total number of turns in all games in the class.

- \(P(s_n|c)\): The probability of observing the \(n\)-action sequence \(s\) given the class label \(c\). This is calculated by finding the number of times that sequence \(s\) appears in games in the class compared to the total number of sequences in all games in the class.
• $P(c)$: The probability of class $c$. This is the number of examples of class $c$ over the total number of training instances.

• $P(i, s_n)$: The joint probability of seeing sequence $s_n$ of on turn $i$. This is defined as $\sum_c P(i|c)P(s_n|c)P(c)$. In other words, it is the sum of the numerator in Equation 4.5 for all values of $c$ — a normalizing factor.

By using the probabilities to compute $P(c|i, s_n)$ for every turn, we can then identify a set of $n$-action sequences that are predictive of the player belonging to a certain class. To do this, the author must define a probability threshold for what is predictive and what is not. This is used to monitor players in-game and make predictions about their class.

### 4.4 Prediction in Scrabblesque

As previously discussed, my technique for modeling session-level player retention consists of the following two steps:

1. Convert baseline analytics into deviation-based analytics
2. Use these analytics to train an $n$-gram model of session-level retention

In the next section, I will describe how these two steps are implemented Scrabblesque. Later in the chapter, I will discuss how these steps are implemented in my other research game, Sidequest: The Game.

#### 4.4.1 Deviation-Based Analytics in Scrabblesque

Earlier, I discussed an experiment that I performed using 5 vanity analytics gathered in Scrabblesque as inputs into various classification algorithms. I have chosen to use the same 5 analytics as the input into the $n$-gram technique for modeling retention. Recall that these 5 analytics are:

• Score difference
• Turn Length
• Word Length
In order to convert these analytics into deviation-based analytics, I applied Equation 4.3, which was introduced earlier in this chapter. The result of this process is a number that represents how much each analytic deviates from its average value at each time step.

Once this is done, I then discretize the data into 3 equal-sized bins corresponding to high values, medium values, and low values to make the probability calculations associated with the n-gram model simpler. Analytics that have a value of “low” are analytics that display low deviation from their mean on a given turn, whereas a “high” analytic value represents a large amount of deviation from that analytic’s mean on a given turn.

4.4.2 The n-Gram Model in Scrabblesque

Once the baseline analytics have been transformed, they are used as training data for an n-gram model used to solve Equation 4.5, which predicts the probability that a player belongs to a certain class. In this research, I am using player class to represent whether a player did or did not perform a specific action, specifically whether or not a player ended a game of Scrabblesque prematurely. So, one can think of this as a binary classification problem with the following classes: 1) The player finishes the game and 2) The player quits the game early.

Also recall that the n-gram model uses action sequences of length n to predict player class. In Scrabblesque, I have chosen to use sequences of length 3 to make these predictions. The reason that I chose 3 is to avoid any potential issues with data sparsity. To illustrate the type of issues that I wish to avoid, let us consider a 10 turn game of Scrabblesque. If $a_i(p, j)$ (the value of analytic i for player p on turn j) can only take on three values, it would still take on the order of $3^{10} = 59,049$ games to explore the space of possible configurations for a 10 turn game. To alleviate this, I do not consider a player’s entire turn history and instead, consider just the last three turns. By choosing 3, I balance predictive power with the problem of data sparsity. If I examine the last two actions, I lose predictive power since I have fewer previous actions that serve as evidence, but I have very few data sparsity issues (it only requires on the order of $3^2 * 9 = 81$ games to explore a 10 turn game). If I examine the last four actions, then I gain predictive power since I have more observations, but increase data sparsity issues (requires on the order of $3^4 * 7 = 567$ games to explore a 10 turn game).
For each feature, the probability $P(c \mid i, s_n)$ is calculated for each turn and is then used to try to predict the class of the player during the current turn. What I end up with is a set of 3-action sequences with an associated probability of whether the player will end the game early. As mentioned earlier, this technique requires a probability threshold to determine if a sequence is predictive or not. In Scrabblesque I use the a priori probability of correctly classifying a player in our dataset, which was 0.241. Therefore, if a sequence is able to perform better than this threshold, then it allows us to make predictions with a better probability than that of a random guess and is, therefore, considered to be predictive of session-level player retention.

### 4.4.3 Experimental Methodology

To evaluate the power of deviation-based analytics as well as the $n$-gram model of session-level retention, I used the Scrabblesque dataset that I described earlier to generate my features and create my model. To summarize, this dataset contains 195 game traces from 94 different users. Of those game traces, 148 traces are completed, whereas 47 traces were not completed.

Using this dataset, I calculated analytic values for each of the 5 analytics that I have described above and then converted them into deviation-based analytics using the SAX transformation. These features were then used to calculate $P(c \mid i, s_n)$ for every action sequence of length 3 on every applicable turn. This means that probability calculations began once I have seen at least 3 actions from the player. Once that probability has been calculated for all 3-action sequences, I used the probability threshold for each turn to filter out all non-predictive sequences.

Once this was done, I analyzed the resulting sequences for their predictive power and for patterns that exist across sequences. The results of these analyses are given in the next section.

### 4.4.4 Results and Discussion

I found several sequences of actions for each of the five deviation-based analytics that are informative as to whether a player will end a game early or not. I calculated the percentage of a game that each player remained in a warning state, a state in which their probability of ending the game prematurely was higher than the predictive threshold value, for each
Table 4.3 Average percentage of a game spent in warning state for each feature in Scrabblesque.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Finished</th>
<th>Unfinished</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Difference</td>
<td>18.8%</td>
<td>43.2%</td>
</tr>
<tr>
<td>Turn Length</td>
<td>30.0%</td>
<td>50.8%</td>
</tr>
<tr>
<td>Word Length</td>
<td>22.2%</td>
<td>51.7%</td>
</tr>
<tr>
<td>Word Score</td>
<td>17.4%</td>
<td>49.2%</td>
</tr>
<tr>
<td>Word Submitted</td>
<td>32.2%</td>
<td>65.2%</td>
</tr>
</tbody>
</table>

Figure 4.1 Probability the player will quit playing Scrabblesque based on the score difference feature. The short dotted line is the baseline, the solid line is the probability of the player eventually quitting at each turn for an example finished game, and the long dotted line is the probability of the player eventually quitting at each turn for an example unfinished game.
Table 4.4 Number of sequences with $P(c|i, s) > 0.241$ in the beginning, middle, and end of a game of Scrabblesque. Beginning is the first five turns, middle is six through ten, and end is 11 and on.

<table>
<thead>
<tr>
<th></th>
<th>Beginning</th>
<th>Middle</th>
<th>End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Difference</td>
<td>45</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>Turn Length</td>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Word Length</td>
<td>46</td>
<td>3</td>
<td>44</td>
</tr>
<tr>
<td>Word Score</td>
<td>53</td>
<td>20</td>
<td>19</td>
</tr>
<tr>
<td>Word Submitted</td>
<td>35</td>
<td>7</td>
<td>13</td>
</tr>
</tbody>
</table>

analytic. The results for that set of experiments can be seen in Table 4.3 for each analytic. The important thing to note is that players who end games prematurely spend much more of their game time in a warning state than players that will finish the game normally—at least twice as long, and all greater than 43% compared to at most 32.2% for players who completed the game. Therefore, the length of time that a player spends in a warning state can also act as a predictor of whether a player will end the current game prematurely. For a better illustration of this, see Figure 4.1. This figure shows how these probabilities change over the course of a game for a given feature (in this case, it is the score difference feature). Notice that for the finished game in Figure 4.1 $P(c|i, s)$ never rises above the threshold value whereas the probability in the unfinished game in Figure 4.1 stays above the threshold for a significant amount of time. These examples were chosen to be illustrative—not all features or games were as clear cut (as indicated in Table 4.3).

These sequences will be most useful if they are able to identify games that are likely to end prematurely as early in the game as possible. This allows ample time for any adaptations that are to be made to have their desired effect. Table 4.4 shows the distribution of sequences based on the turn that they occur on. As you can see, most sequences associated with players quitting the game occur in the beginning of a game; however, the word length feature and the word submitted feature display bimodal behavior. In Figure 4.2a I show the word submitted occurrence histogram in more detail and in Figure 4.2b I show the word length occurrence histogram in more detail. As you can see, both of these have a high concentration of sequences found at the beginning and the end of the game with very few sequences found in the middle of the game. This implies that the characteristics of players’ play early and late in the game are the most important when it comes to determining they will end the game prematurely. The explanation for this behavior at the beginning of the game is likely because the player has not invested enough time into the game. Thus, it is
Figure 4.2 Turn histograms for the word submitted and word length analytics in *Scrabblesque*. This shows how many predictive sequences were found on each turn.
Table 4.5 Percentage of values at each position in informative action sequences for each feature. Position 1 represents the earliest event looked at while Position 3 represents the most recent event looked at.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Position 1 Low</th>
<th>Position 1 Med</th>
<th>Position 1 High</th>
<th>Position 2 Low</th>
<th>Position 2 Med</th>
<th>Position 2 High</th>
<th>Position 3 Low</th>
<th>Position 3 Med</th>
<th>Position 3 High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Diff</td>
<td>30.3%</td>
<td>37.9%</td>
<td>31.8%</td>
<td>33.3%</td>
<td>31.8%</td>
<td>34.9%</td>
<td>10.6%</td>
<td>42.4%</td>
<td>47.0%</td>
</tr>
<tr>
<td>Turn length</td>
<td>72.7%</td>
<td>27.3%</td>
<td>0.0%</td>
<td>45.5%</td>
<td>18.2%</td>
<td>36.3%</td>
<td>27.4%</td>
<td>36.3%</td>
<td>36.3%</td>
</tr>
<tr>
<td>Word Length</td>
<td>28.0%</td>
<td>53.7%</td>
<td>18.3%</td>
<td>32.3%</td>
<td>37.6%</td>
<td>30.1%</td>
<td>39.8%</td>
<td>23.7%</td>
<td>36.5%</td>
</tr>
<tr>
<td>Word Score</td>
<td>26.1%</td>
<td>26.1%</td>
<td>47.8%</td>
<td>17.4%</td>
<td>54.3%</td>
<td>28.3%</td>
<td>28.3%</td>
<td>41.3%</td>
<td>30.4%</td>
</tr>
<tr>
<td>Word Submit</td>
<td>71.0%</td>
<td>20.0%</td>
<td>9.0%</td>
<td>36.4%</td>
<td>38.2%</td>
<td>25.4%</td>
<td>18.2%</td>
<td>38.2%</td>
<td>43.6%</td>
</tr>
</tbody>
</table>

possible that the player finds it easier to leave the game since they are not too invested in it or its outcome. At the end of the game, this behavior is likely caused by the outcome of the game already being decided. If either the player or the AI has a large lead, players may feel that ending the game is not necessary since the outcome has already been decided.

Finally, I sought to draw general conclusions about the sequences that are predictive of players quitting. A summary of the composition of the sequences is in Table 4.5. Note that there are two types of analytics observed here: those correlated with score and those based on time. The score difference, word length, and word score analytics are correlated with the player’s performance. For the word length and word score analytics, predominately high deviations in position one and lower deviations in positions 2 and 3 indicate an increased likelihood of quitting. Given that, it makes sense that the opposite is true for score difference. If lower score words are submitted, the score difference feature is likely to increase. When considering the time-based features turn length and word submitted, I see a strong trend from low deviations towards high deviations from average being predictive of quitting. Notice that for each of these features, position 1 values are typically low and they move towards the high value in positions 2 and 3. In this case, players are likely either turned away by the higher cognitive load as the board fills in, or become distracted and less interested in playing.

### 4.5 Prediction in SQ:TG

As mentioned previously, SQ:TG represents a significant step up in complexity from Scrabblesque. The player has more flexibility in how they choose to interact with the environment,
there different activities to perform, and there is no clear “opponent” that the player is competing against. In the following sections, I detail how I used the set of vanity analytics discussed in the previous chapter to model session-level retention.

### 4.5.1 Analytics in Sidequest: The Game

In SQ:TG, accepting and completing quests is the core game mechanic. While each quest has different tasks that the player must complete, the game, in general, is about selecting the quests that you want to complete and then finishing them in order to progress to the next stage. As such, I chose to use the following 2 analytics describing player interactions with quests to model session-level player retention:

- Action Type
- Action Times

As these analytics have been discussed previously, I am listing these only to remind the reader of the analytics being used. For a more detailed discussion and definition of these analytics, please refer to Chapter 3. It is important to note that I did not take into account the specific quest that was involved in any of the action types. The reason for this is that I wanted to avoid/mitigate the effects of the curse of dimensionality. If I had considered each quest available when considering action type, then each action type would have 30 possible values (10 in each stage). Thus, I would require enough data to observe the $4 \times 30$ possible actions available to the player (4 action types, each type taking on 30 possible values). Gathering this amount of data is very difficult given the amount of resources at my disposal and, as such, I chose to think of action types in abstract.

One difference to note between the analytics chosen for SQ:TG and those chosen for Scrabblesque is that only one of them is numeric (action times). This means that only the analytic describing action time needs to undergo the SAX transformation and subsequent discretization. Thus, only the time features associated with each quest action were converted to a set of deviation-based features by using Equation 4.3 and then discretized into 3 equal-sized bins corresponding to high, medium, and low, as was done with the features in Scrabblesque. Recall that this means that the timing features in SQ:TG are based on how much they deviate from its expected value. This means that if the time it takes to complete a quest is low, then the time it took to complete that quest was close to the average time
it takes to complete a quest. If the time it took to complete a quest is high, however, that means that it took much longer (or shorter) to complete this quest than was expected.

### 4.5.2 The n-Gram Model in Sidequest: The Game

Once these analytics have been transformed, they are used to train an $n$-gram model to calculate Equation 4.5. As with Scrabblesque, the class that I am trying to predict is whether or not a player quits the game before it would normally end. Ideally, model building would proceed as it did in Scrabblesque. I would calculate $P(c|i, s_n)$ for each $n$-action sequence on each turn in the game. If $P(c|i, s)$ is greater than the a priori probability of correctly predicting if a player ended the game early, then the sequence $s$ is considered to be predictive of a player quitting the game early. That being said, there are some aspects of SQ:TG are problematic when performing this analysis. The first of these is the notion of a turn in SQ:TG. Recall that Scrabblesque consists of the human player and a computer-controlled AI taking turns playing words onto a game board. It is intuitive to divide up a game of Scrabblesque into discrete time events based on turns. In SQTG, there is not a definitive notion of a turn, which makes it unclear how best to divide up a game of SQ:TG.

In addition to the difficulty in defining a turn in SQ:TG, it is also not clear how each of the analytics I consider should be treated. If I use these analytics as I did in Scrabblesque, then I would treat them as being independent and use them to construct separate models. This is fine for the analytics that describe action type, but considering action timings as independent from the actions they describe seems to be fundamentally flawed. The other option is to model them together such that each action has a timing value associated with it. This makes it very difficult to assemble enough examples to fully explore the space of possible sequences. For example, consider a 3-action window where I use both action types as well as action timings to construct the models. Since each action taken will have a time associated to it, we can consider a single action having 12 possible values. This means that a 3 action window would result in 1728 possible combinations of action type and action time.

There is also the question of what the best value of $n$ is for this problem. In Scrabblesque, there was some reasoning about choosing $n = 3$, but it was never determined to be the best $n$ value to use in that domain.

To resolve these issues, I consider several possible parameter values (number of turns, which analytics to use, and the value of $n$) and then perform an experiment to determine
which configuration produces the best models of session-level retention. Those models will then be used as inputs into my technique for generating dynamic game adaptations which will be discussed in greater detail in Chapter 5.

4.5.3 Experimental Methodology

There are two main goals of these experiments:

1. To determine the set of parameters that produce sequences of vanity analytics with the most discriminative power with respect to session-level retention

2. To identify a set of sequences that are predictive of session-level retention for use in dynamically adapting SQ:TG

The most important of these goals is to determine what are the optimal parameter values for the \( n \)-gram model are. Recall that I am considering 3 different types of parameters each with several different values. To summarize, they are:

- Sequence Length: The number of actions \( n \) that compose a sequence \( s_n \) for calculating \( P(c|i, s_n) \). It consists of two possible values, which are sequences of size 2, and 3.

- Game Turns: How many turns are in a game. It consists of two possible values, which are 9 (turns end when a quest is completed) and 3 (turns end when a stage is completed).

- Types of Features: The types of features that compose a sequence \( s \). It consists of three possible values, which are action types by themselves, action timings by themselves, and action types paired with their associated action time.

In these experiments, I test every one of the possible parameter configurations, meaning that each one is a different condition. This means that there are a total of 12 parameter configuration that I am testing \((3 \times 2 \times 2 = 12)\). Once this parameter configuration has been chosen, then accomplishing the second goal is simply a matter of using that parameter set to create a predictive set of sequences.

To test this, I made SQ:TG available to the public online and actively recruited people for a period of about one month. I used many different social media outlets, forums, and mailing lists to recruit participants. For the purposes of all analyses related to SQ:TG, I
only consider games in which the player completed at least one quest. This is done to help ensure that the game traces recorded show players at least attempting to complete the game rather than just superficially exploring the game world or the game mechanics. At the end of the data collection period, I had collected 266 game traces, of which 141 were completed and 125 were incomplete. A summary of these games can be seen in Table 4.6.

Once data collection was complete, I extracted the desired analytic values from each player’s total game trace. The result of this step was a trace of player actions (accepting, rejecting, abandoning, or completing a quest) and the time associated with each action. Using these traces, I calculated \( P(c|i,s_n) \) under each experimental condition mentioned earlier. Once this has been done, sequences are filtered based on the \textit{a priori} probability of predicting player class on a particular turn.

Since one of the goals of this experiment is to find the set of parameter values that provides the most discriminative power between classes, I evaluate the percentage of the game that is spent in \textit{warning state}. Recall that a player is in a warning state if their last \( n \) actions were predictive of them quitting the game early. 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:

\[
d_n = w_i - w_c
\]  

In other words, we calculate the discriminative power, \( d_n \), associated with a set of sequences, \( d_n \), of length \( n \), 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, \( w_i \). If \( d_n \) is high, then I consider that set of sequences to have a high amount of \textit{discriminative power}. The results of this analysis is given in the next section.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
 & Games Played & Players \\
\hline
Finished & 141 & 141 \\
Unfinished & 125 & 122 \\
\hline
\end{tabular}
\caption{Summary of \textit{Sidequest: The Game} dataset. Games played shows the number of games played in each category. Players shows the number of unique players in each category.}
\end{table}
Table 4.7 Percentage of time spent in warning state when completing a quest marks the end of a turn across all conditions. Rows in bold indicate that differences found in this condition were statistically significant ($p < 0.05$) according to a two-tailed, independent samples T-Test.

<table>
<thead>
<tr>
<th>Feature Types</th>
<th>Sequence Length</th>
<th>Completed</th>
<th>Incomplete</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Types</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Action Sequences</td>
<td>27.0%</td>
<td>45.2%</td>
<td>18.2%</td>
<td></td>
</tr>
<tr>
<td>3-Action Sequences</td>
<td>22.8%</td>
<td>40.6%</td>
<td>17.8%</td>
<td></td>
</tr>
<tr>
<td><strong>Action Times</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Action Sequences</td>
<td>29.5%</td>
<td>31.1%</td>
<td>1.6%</td>
<td></td>
</tr>
<tr>
<td>3-Action Sequences</td>
<td>34.3%</td>
<td>29.0%</td>
<td>-5.3%</td>
<td></td>
</tr>
<tr>
<td><strong>Combination</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Action Sequences</td>
<td>39.1%</td>
<td>47.1%</td>
<td>8.0%</td>
<td></td>
</tr>
<tr>
<td>3-Action Sequences</td>
<td>11.0%</td>
<td>19.3%</td>
<td>8.3%</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.8 Percentage of time spent in warning state when completing a stage marks the end of a turn across all conditions. Rows in bold indicate that differences found in this condition were statistically significant ($p < 0.05$) according to a two-tailed, independent samples T-Test.

<table>
<thead>
<tr>
<th>Feature Types</th>
<th>Sequence Length</th>
<th>Completed</th>
<th>Incomplete</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Types</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Action Sequences</td>
<td>22.2%</td>
<td>47.0%</td>
<td>24.8%</td>
<td></td>
</tr>
<tr>
<td>3-Action Sequences</td>
<td>20.3%</td>
<td>43.2%</td>
<td>22.9%</td>
<td></td>
</tr>
<tr>
<td><strong>Action Times</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Action Sequences</td>
<td>59.7%</td>
<td>63.6%</td>
<td>3.9%</td>
<td></td>
</tr>
<tr>
<td>3-Action Sequences</td>
<td>46.3%</td>
<td>51.3%</td>
<td>5.0%</td>
<td></td>
</tr>
<tr>
<td><strong>Combination</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-Action Sequences</td>
<td>13.9%</td>
<td>34.1%</td>
<td>20.2%</td>
<td></td>
</tr>
<tr>
<td>3-Action Sequences</td>
<td>4.3%</td>
<td>17.1%</td>
<td>12.8%</td>
<td></td>
</tr>
</tbody>
</table>

### 4.5.4 Results and Discussion

The results of this analysis have been compiled into two tables. Table 4.7 shows the results of the warning state analysis when I consider turns ending after completing a quest. As you can see in the table, incomplete games tend to spend much more time in a warning state compared to completed games across all conditions except when we only consider the times of actions. For 9-turn games, both 2-action and 3-action time sequences did not produce any significant differences between complete and incomplete games. In the case of 3-action sequences, completed games actually spent more time in warning state than incomplete games.

The results of the warning state analysis when I consider turns ending after completing a stage are shown in Table 4.8. These results are similar to those found with the 9-turn analysis. When I use only action type features or a combination of action types and action...
Table 4.9 The number of action sequences that are predictive of players quitting during each stage of Sidequest: The Game

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>5</td>
<td>8</td>
</tr>
</tbody>
</table>

times, I find that incomplete games spend a significantly more time in warning state than complete games.

What is interesting to note here is that the differences in warning state percentages for this turn representation are, in general, higher in the 3-turn representation than those for the 9-turn representation. This is likely due to the disparity in the amount of observational data that is available under the different conditions. Simply put, the 9-turn representation of SQ:TG contains fewer actions *per-turn* than the 3-turn representation. This means that the \( n \)-gram model simply has more data that it can use in the 3-turn representation than the 9-turn representation.

By examining the tables, it becomes clear that the condition that had the best performance was using the 3-turn representation with 2-action sequences consisting of only action types. From here on, it is assumed that I am using these vanity analytics to build models and implement game adoptions.

Analysis of the sequences produced provides some interesting insights into what contributes to session-level player retention. Table 4.9 shows how many sequences were deemed predictive of players quitting early for each act. It is interesting to note that there are more sequences during the first stage of the game and the last stage of the game than during the middle. This implies that player exploration through the space of possible actions is likely to be discouraged during these two stages of the game; however, during the middle of the game the player has more of an opportunity to explore the space of actions to complete while still maintaining a low probability to quit the game. Recall that in *Scrabblesque* there were certain analytics that implied that the beginning and the end of the game were the most important times of the game for predicting session-level retention. The results in Table 4.9 reinforce this statement by implying that more control will be necessary at the beginning and end of the game in order to get players to complete a game session.

In addition to this analysis, I also measured how often each action appeared in a predictive sequence. Table 4.10 shows the results of this analysis. I performed this analysis by dividing the number of times an action is observed by the total number of actions across
Table 4.10 Percentage of time that an action occurs in a predictive sequence for each act. Percentages are calculated as the number of times an action did appear divided by the number of times that it could appear.

<table>
<thead>
<tr>
<th></th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quest Accepted</td>
<td>33.3%</td>
<td>30.0%</td>
<td>25.0%</td>
</tr>
<tr>
<td>Quest Rejected</td>
<td>38.9%</td>
<td>40.0%</td>
<td>43.8%</td>
</tr>
<tr>
<td>Quest Abandoned</td>
<td>22.2%</td>
<td>30.0%</td>
<td>18.7%</td>
</tr>
<tr>
<td>Quest Completed</td>
<td>5.6%</td>
<td>0.0%</td>
<td>12.5%</td>
</tr>
</tbody>
</table>

all sequences and reporting the result as a percentage. In this case, each sequence contains 2 actions since that was one of the parameter values that obtained the most discriminative power. The main thing to take away from the table is that rejecting a quest appears more often than any other action in these sequences. This implies that the act of rejecting a quest is very indicative of session-level retention on its own. Also notice that the percentages associated with accepting a quest are quite high. This could be an anomaly since you are required to accept quests to advance through the game; however, it could also indicate that accepting more quests than are required (through failing your current quest or abandoning your current quest) is associated with a player quitting the game early.

4.6 Conclusion

In this chapter, I have shown how I use a set of vanity analytics to model session-level retention in both Scrabblesque and SQ:TG. When applicable, raw analytic values were converted into deviation-based analytic values using a SAX transformation and were then used as inputs into an n-gram model. Using this model, I calculated the probability that each player would quit the current game early based on their past actions. The result of this is a set of sequences that are predictive of players ending each game before it would naturally conclude. I have also presented an analysis of the resulting sequences for each game environment and given some insight as to what certain sequences mean and why certain analytic values behave the way that they do. Now that these sequences have been generated, they will become the basis for my dynamic game adaption technique for improving session-level player retention, which will be discussed in greater detail in the next chapter.
5.1 Introduction

In the previous chapter, I discussed how vanity analytics can be used to model session-level retention in both Scrabblesque and Sidequest: The Game. These models, however, are not very useful with regards to dynamic game adaption since they are expressed in terms of analytics that are not directly affectable by an AI system. This is another example of the gap between vanity and actionable analytics. In order to use the models that I have created to improve player retention, I must first find a way to bridge the gap between vanity and actionable analytics. In other words, I need to find a way to use the models that I have created to determine how to alter the actionable analytics that I can actually control. In terms of the political metaphor that I have been using throughout this document, I need to use the results of the polls to determine how best to distribute my money.

In this chapter, I outline my technique for utilizing player models created with vanity analytics in order to create an adaptive game environment. In my case, I choose to use an experience manager. The term experience manager was coined by Mark Riedl who defines it
as a generalization of a drama management (2008). In other words, an experience manager is an autonomous agent that monitors player behavior and then makes alterations to the game world in order to encourage certain play behaviors or to enhance some aspect of the current player’s gameplay experience. I will outline the details of how my experience manager utilizes these player models to make adaptions to the game world using data-driven backwards induction for action generation (DDBIAG).

5.2 Motivation

One of the key insights introduced in this chapter is that I could use backwards induction (Von Neumann and Morgenstern, 2007) to generate actions in real-time such that an experience manager could elicit a desired behavioral response from players. Backwards induction is a technique that is used to determine an optimal sequence of actions by determining what action should be done immediately preceding the goal state and then recursively working backwards towards the current state until a solution is found. What is especially appealing about this technique is that we can select a several different goal states in order to increase the variability of replays. I consider this an improvement over other approaches to experience management, such as MDP-based approaches, that find the “best” sequence of actions to achieve a goal. These approaches can only produce a limited number of possible experiences since they are designed to find the best sequence of actions to achieve a goal. In contrast, my approach intends to select many different goals from a distribution of goals that result in above-average experiences in order to increase replayability. In this case, I consider the ability to produce a variety of above-average gameplay experiences more important than the ability to produce the best gameplay experience.

Furthermore, the DDBIAG algorithm needs to be able to bridge the gap between vanity analytics and actionable analytics in order to dynamically adapt a game world. The DDBIAG algorithm does this by using the models created by examining vanity analytics to determine how to affect a set of actionable analytics in order to bring about the desired change in player behavior. At a high level, the DDBIAG algorithm uses the player models to find a set of game states that are predictive of players quitting the game early and then represents these states in terms of actionable analytics.
5.3 The DDBIAG Algorithm

As mentioned earlier, my technique for experience management depends on a data-driven backwards induction algorithm to generate actions for an experience manager to perform. In this context, one can think of the actions that the experience manager will take as adaptions that the experience manager will introduce to the game environment.

At a high level, backwards induction consists of the following steps:

1. Select goal state
2. Generate candidate actions
3. Select action that minimizes distance to the goal state

In order to translate this general framework into a game adaption scheme, several questions needed to be addressed. How do we represent the goal state? How do we determine distance in this environment? What specific actions can be performed? These are examples of the types of questions that needed to be addressed. In the end, I expanded the technique in order for it to be used for game adaption. The steps involved in the data-driven backwards induction algorithm for action generation are listed below:

1. Select a goal state
2. Retrieve game states based on this goal
3. Generate candidate actions and resultant candidate game states
4. Calculate distance between candidate game states and the goal state
5. Select the action that minimizes the distance between candidate game states and the goal state

A visualization of this process can be seen in Figure 5.1. In the following sections, I will discuss the execution of each of these steps in greater detail.
5.3.1 Select a Goal State

The first step of my algorithm is to select a goal state. For the purposes of this technique, I have chosen to represent goal states in terms of the behaviors that I want players to exhibit. The specifics of these behaviors, however, is determined by the player model that is used as an input into this algorithm. For example, if we are using a player model that predicts the number of deaths a player will experience in a game level, the goal state would refer to the number of deaths that I want the player to experience.

In order to increase the number of possible experiences that my algorithm can present to the player, the goal state is selected in accordance to a target distribution of possible goal states. Using the above example, my algorithm would select the number of deaths that it wants the player to experience from a distribution of death counts that has been authored according to the player model beforehand. In other words, the player model is converted into this target distribution of goal states. There are many ways to author a target distribution (Roberts et al., 2007), but it is beyond the scope of this paper to go over techniques used to generate these distributions.

5.3.2 Retrieve Game States

Once the algorithm has selected a goal state, it needs to figure out the best way to achieve this goal state. Since this is a data-driven algorithm, this is done by examining player data
that has already been recorded. Specifically, this step of the algorithm involves retrieving game states that are associated with the goal state that we wish to achieve. So, if we want the player to have 5 deaths on a certain level, this stage of the algorithm will involve retrieving all game states in which that player had 5 deaths.

What exactly do I mean when I say game state? This is another representational issue that largely depends on the author. What is important is that game states need to be represented in terms of actionable analytics. This means that a set of actionable analytics will need to be calculated for every game state that was retrieved in this step. In the above example, possible actionable analytics include the number of enemies that are in the level or the number of traps that are in the level. So, one can think of this step as retrieving a set of actionable analytic values that describe the goal state.

5.3.3 Generate Candidate Actions and Resultant States

The next step is to generate a set of candidate actions and simulate their effect on the game state. The exact actions that the experience manager can take depend on how much control the experience manager can feasibly have over the game world. That being said, the actions taken will always cause the values of the set of actionable analytics to change in some foreseeable way. To continue with the example used above, the actions taken in this case could be to place enemies/traps in the game world. Based on how many enemies/traps are placed and where they are placed, any actionable analytics related to the number of traps and their location will be changed.

For every candidate action generated, its effect on these actionable analytics should be measured. Typically, this involves simulating the completion of the action and then recalculating all actionable analytics. This means that our system would place enemies and traps in my example scenario and then recalculate the actionable analytics based on this new game world. For every action, these features are then saved into a set of candidate game states.

5.3.4 Calculate Distance Between Candidate Game States and Goal State and Select Action

The experience manager then calculates the distance from each candidate game state that resulted from a candidate action to the goal state. The experience manager selects the
action that produced the candidate game state that minimizes the distance between it and the target game state. It is important to note that there is no restriction on the type of distance metric that is used. So, in our example the experience manager could calculate the Euclidean distance between the actionable analytics associated with the candidate game states and the actionable analytics associated with the goal state.

Since the actions taken during this step are meant to influence the values of a set of actionable analytics, my system does not directly influence player behavior. As in the political metaphor, this system can only attempt to indirectly influence player behavior (the polls) by influencing these actionable analytics (distributing money).

5.4 Implementation in Scrabblesque

In this section, I will outline the details of how my experience management scheme was implemented in Scrabblesque. As I discussed previously, the steps of my algorithm are as follows:

1. Select a goal state
2. Retrieve game states based on this goal
3. Generate candidate actions and resultant candidate game states
4. Calculate distance between candidate game states and the goal state
5. Select the action that minimizes the distance between candidate game states and the goal state

In the following sections, I will discuss the implementation of each of these steps in the testbed game, Scrabblesque.

5.4.1 Select a Goal State

Recall that the first step in the DDBIAG algorithm is to select a goal state from a target distribution of game analytics. Before the goal state can be selected, this target distribution must be authored based on the player model created for the game.

The player model for Scrabblesque (discussed in Chapter 4) consists of 3-actions sequences for each of the following 5 features:
• Score Differential
• Turn Length
• Average Word Score
• Average Number of Words Submitted
• Average Word Length

Recall in Chapter 4 that I discovered that the turn length feature was not informative. For the purposes of this implementation, it is no longer considered part of the player model for determining the probability that a player will quit a game of Scrabblesque early. Therefore, this player model is considered to consist of only 4 features. So, moving forward these are the behaviors that I am seeking to influence. It can be argued that I actually want to influence play duration in Scrabblesque and not these features. As I showed in Chapter 4, certain sequences of feature values are predictive of ending the game early. The natural assumption is that by changing these features, I can bring about this desired change in play time.

So before goal selection can occur, this model must be transformed into a target distribution. As previously stated there are many strategies for this. For this implementation, I have chosen to use Monte Carlo Sampling using the Metropolis-Hastings algorithm (Hastings, 1970). 10,000 games of 20 turns each were simulated and then evaluated to create a target distribution over feature values for each of these 4 gameplay features.

When the experience manager selects a goal state from this distribution, it is selecting a 4-dimensional vector with each dimension corresponding to a specific gameplay feature.

5.4.2 Retrieve Game States

The next step is to retrieve a set of game states that match the goal state. Due to the turn-based nature of Scrabblesque and the nature of the analytics used to describe the goal state, this step presented some unique difficulties. First, the features that we are seeking to influence are based on actions taken during the player's turn. Since the experience manager cannot directly affect what moves the player will make, the experience manager must return a set of intermediate goal states. In Scrabblesque, an intermediate goal state is the state the board was in on the computer turn immediately preceding the goal state. The intuition
behind doing this is that our experience manager should be able to put the board into these intermediate goal states, which should then influence the player to make the moves to get them into the goal state.

The nature of the analytics describing the game state was also problematic. Recall that I used a set of *vanity* features to model session-level retention, which are not directly affectable by my experience manager. As I said previously, the game states in this step must be in terms of actionable analytics that describe the game state. To address this, I came up with a set of actionable analytics that I discussed in Chapter 3. For reference, the set of actionable analytics that describe the state of the board are:

- Number of Candidate Tiles
- Consonant/Vowel Distribution
- Average Tile Value
- Proximity to Bonus Squares

In addition to these metrics, I also used three others to describe which tiles the player had:

- Consonant/Vowel Distribution
- Average Tile Value
- Number of Repeated Tiles

Once a set of *intermediate goal states* has been generated, each game state is expressed in terms of these analytics. So, the target distribution created using vanity analytics is used to identify ideal board configurations that the game can be put in. By calculating this set of actionable analytic values, my experience manager can bridge the gap between vanity and actionable analytics. Therefore, it is able to use the knowledge contained in the player model of session-level retention to make explicit changes to the environment. For the remainder of the algorithm, consider each *intermediate goal state* to be in terms of these actionable analytics.
5.4.3 Generate Candidate Actions and Resultant States

Next, the experience manager will generate a set of candidate actions randomly and observe what effect they will have on the game world. These resultant game states are referred to as intermediate candidate states. In Scrabblesque, the actions that the experience manager generates are the words that the computer plays and the tiles that the computer gives the player once they have played a word. In my implementation, the computer does not use a rack, and instead has access to the entire bag of remaining tiles. As such, my system can generate many candidate words for the computer to play as it is not limited by a rack. Further, it can give the player specific tiles to make their job of completing words easier or harder.

In Scrabblesque, the experience manager is called whenever the computer-controlled AI plays a word or refills the player's rack. When this occurs, the experience manager will randomly generate candidate actions (either playing a word or selecting tiles to give the player) for a set amount of time. Specifically, the experience manager will generate candidate actions for 1.5 seconds. During this time, the experience manager is also simulating the move and calculating the game analytics that I have mentioned earlier. The intermediate candidate states, as with the intermediate goal states, are expressed in terms of these directly affectable game analytics.

5.4.4 Calculate Distance Between Candidate Game States and Goal State and Select Action

The experience manager then calculates the distance from each intermediate candidate state to each intermediate goal state. The experience manager then selects the action that produced the candidate game state that minimizes the distance between it and the target game state.

In Scrabblesque, I calculate the average Euclidean distance for each candidate action between all target game states. The experience manager then selects the action that minimizes this average distance. Here, I use Euclidean distance because all analytics are numeric, and Euclidean distance is a simple distance formula to use in that situation. In general, any arbitrary distance or divergence measure like KL-divergence or Bhattacharyya divergence (Bhattacharyya, 1943) can be used.
5.5 Implementation in SQ:TG

In the following sections, I will detail the implementation of my experience management scheme in Sidequest: The Game.

5.5.1 Select a Goal State

As discussed earlier, the first thing that needs to be done is to select a goal state from a distribution of game states to target. In Chapter 4, I concluded that the best features to use in order to predict player retention were 2-action sequences of action types. Recall the 4 types of action types that I consider are as follows:

- Accepting a quest
- Rejecting a quest
- Abandoning a quest
- Completing a quest

As with the target distribution for Scrabblesque, the target distribution for SQ:TG was generated by using the Metropolis-Hastings algorithm (Hastings, 1970). 10,000 completed games were generated and then evaluated to create this distribution. In SQ:TG, there are certain restrictions on the distributions that can be created. This is because there are constraints put in place by the game mechanics of SQ:TG that make certain distributions impossible. For example, it is impossible to complete two quests in a row without performing any other action first. As a result, this particular sequence will never naturally occur. This means that any distribution in which this sequence has a non-zero probability distribution associated with it is invalid.

For SQ:TG, I chose to create a single target distribution for each stage of the game resulting in a total of 3 target distributions. A summary of these distributions can be found in Figure 5.2.

In order to choose a goal state to target, I use these distributions to generate a sequence of actions that results in the player completing a quest depending on the stage of the game. This action sequence becomes the goal state that I wish to target. Specifically, this is done by converting the target distributions into a set of transition matrices and using those to
Figure 5.2 Target Distributions of Actions for Each Stage in *Sidequest: The Game*
Table 5.1 Transition Matrices for *Sidequest: The Game*

(a) Transition Matrix for Stage 1

<table>
<thead>
<tr>
<th>First Action</th>
<th>Second Action</th>
<th>Accept</th>
<th>Reject</th>
<th>Abandon</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Reject</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Abandon</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

(b) Transition Matrix for Stage 2

<table>
<thead>
<tr>
<th>First Action</th>
<th>Second Action</th>
<th>Accept</th>
<th>Reject</th>
<th>Abandon</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>0.03</td>
<td>0.33</td>
<td>0.0</td>
<td>0.64</td>
<td></td>
</tr>
<tr>
<td>Reject</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Abandon</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

(c) Transition Matrix for Stage 3

<table>
<thead>
<tr>
<th>First Action</th>
<th>Second Action</th>
<th>Accept</th>
<th>Reject</th>
<th>Abandon</th>
<th>Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accept</td>
<td>0.03</td>
<td>0.0</td>
<td>0.0</td>
<td>0.97</td>
<td></td>
</tr>
<tr>
<td>Reject</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Abandon</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
<tr>
<td>Complete</td>
<td>1.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>
generate a set of target actions. The transition matrices used are shown in Table 5.1. These matrices give a greater insight into what, according to my models, must be done in order to improve player retention. According to Table 5.1a, all action sequences should consist of a player accepting a quest and then immediately completing it. This could be because players are more likely to quit the game if presented with activities that they do not want to complete early in the game. By examining, Table 5.1b, we see that players, ideally, should explore a little more in Stage 2. According to the transition matrix, there is some allowance for players to reject quests instead of immediately completing them. This likely is because players, at this point, have invested a fair amount of time into the game and are less likely to abandon it if they find that there are not many activities that they want to complete. In Stage 3 (Table 5.1c), the goal shifts back towards having players accept and then immediately complete quests. This is likely because it is near the end of the game and players would be more interested in completing the game than finding specific quests to complete.

Once this sequence has been generated, it must be used to retrieve a set of game states.

5.5.2 Retrieve Game States

In SQ:TG, the player is free to complete any quest available in a stage, they just have to physically go to the quest giver and then accept it. In this environment, finding common game states that involve specific quest giver placement in the environment is problematic. This is because (as was discussed in Chapter 1) quests are randomly assigned to quest givers. This means that finding common game states involving specific NPCs or their placements is unlikely. Instead, I chose to think of game states in SQ:TG in terms of quests that players interacted with. The game state for a player consists of the set of quests that said player interacted with and how they interacted with those quests (accepting the quest, rejecting the quest, etc).

In order to retrieve these states, my algorithm finds the k-nearest neighbors (with k = 5, chosen arbitrarily) to the player’s current game state. 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.

At the end of this process, my algorithm has the set of the 5 most similar game states to the current player’s game state.
5.5.3 Generate Candidate Actions and Resultant States

The next step is to generate a set of candidate actions and observe how they will affect the game world. Before I discuss how this is done, let us first consider what exactly an action is in SQ:TG. Given the models created (as discussed in Chapter 4), it makes sense for the actions available to the experience manager to influence which quests a player accepts, rejects, abandons, and completes. In SQ:TG, the only action available to the AI that could perform this task is deciding which NPC gives out which quest. Therefore, the only action available to the experience manager is the ability to control, in real time, which quests can be accepted from NPCs.

Candidate actions are chosen by considering which types of quests need to be presented to the player. For example, consider the case where the experience manager wants the current player to accept a quest and immediately complete it. If the player has not currently accepted a quest, then we want to present them with quests that they are likely to complete. In this case, the system's actions consist of placing quests that the player is likely to accept and complete closer to the player. This is done by finding quest-giving NPCs that are close to the player in the virtual environment and having them give out certain quests that the player is likely to complete. If that player has accepted a quest, however, the system should not offer the player quests that they will likely accept. This is because the player may be tempted to abandon their current quest and accept the new quest. In order to minimize the number of extra actions taken by the player, the experience manager should now offer the player quests that they are likely to reject. In this way, the player is more likely to complete the original quest they accepted. Typically, the system need only look at the current goal action to determine which types of quests to present to the player. So if the player needs to reject a quest, then the system should present players with quests they are likely to reject.

Candidate actions are generated by determining which types of quests need to be presented to the player (using the method described above) and assembling a set of candidate quests that could be presented to the player. The candidate set of quests is taken from the set of 5 similar game states that was retrieved in the previous step. If the player needs to be presented with quests that they need to accept, then the candidate set of actions is the set of quests that were accepted in the game states retrieved earlier. Similarly, if the player needs to be presented with quests that they need to reject then the appropriate set of quests is returned from the retrieved game states and so on.
5.5.4 Calculate Distance Between Candidate Game States and Goal State and Select Action

The final step is to determine which action will best bring about the goal state. In SQ:TG, this means that we have to select quests and assign them to quest-givers such that it is likely that the players will transition into the goal state. The first part of this involves somehow calculating the distance between candidate game states and the goal game state. In SQ:TG, this is not as straightforward as it was in Scrabblesque since game states in SQ:TG do not contain numeric values. Therefore, my experience manager uses a slightly different metric to determine which actions to take. Since goal states consist of actions such as accepting or rejecting a quest, the system will present quests to the player that they are likely to accept, reject, abandon, or complete depending on what is needed to satisfy the goal.

So now, the action that moves us 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. This action is selected by finding at quests that occur most often in the game states retrieved earlier. If the player should abandon the quest that they are currently on, for example, then the system will look for which quests are commonly abandoned in the retrieved game states and select those for placement. If there are multiple quests that occur with the same frequency in the retrieved game states, then they are chosen from at random. Quests are placed in the world based on their proximity to the player. So, quests that are likely to result in a goal action are given to quest-givers that are closer to the player.

This process repeats until all quest-givers that are visible have a quest to hand out. If a player has already talked to a quest giver, then that quest is considered locked and can no longer be moved. This effectively removes it from placement consideration, which means that it will be skipped over if it would have been placed during this step. In terms of actionable analytics, the experience manager attempts to minimize the distance between quests that the player is supposed to interact with and the player.

5.6 Conclusion

This chapter contains the details of the DDBIAG algorithm, my algorithm for experience management that bridges the gap between vanity analytics and actionable analytics. It
does this by using the insights gained from vanity analytic models to identify predictive
game states. These game states are then expressed in terms of a set of actionable analytics
describing these game states. It then intelligently chooses actions that will alter these
actionable analytic values with the goal of influencing, albeit indirectly, player behavior. I
have also discussed the implementation of this algorithm in both Scrabblesque and SQ:TG.
In the next chapter, I will discuss the methodology and results of experiments performed
in both of these game environments to improve session-level player retention.
CHAPTER 6

EVALUATING SESSION-LEVEL RETENTION

6.1 Introduction

In this chapter, I will discuss evaluation of the experience manager that has been described in the previous chapter. The evaluation of my system consists of two separate evaluations, one for each game environment. Since the main goal of this experience manager is to improve session-level player retention by indirectly influencing player behavior, I conduct a quantitative analysis to ensure that the behaviors that I want to bring about are actually happening. This translates, in both game environments, into analyzing how well the experience manager is able to realize the desired distributions of player behavior. Since the player model is often designed with a behavioral purpose in mind, I also quantitatively verify that this is taking place. In this research, this is done by analyzing the quitting rate since the behavioral result I am attempting to achieve is an increase in session-level retention.
6.2 Analysis in Scrabblesque

To validate my results, I performed a user study and examined the logs of player data that were produced. In the following experiments, I compare the performance of an adaptive version of Scrabblesque with the performance of the non-adaptive version used to create the models of player retention discussed in Chapter 4. In the following sections, I will refer to the distribution of observations produced by the adaptive version of Scrabblesque as the adaptive distribution, and I will refer to the distribution of observations produced by the non-adaptive version of Scrabblesque as the non-adaptive distribution. This section will go over the results of this study.

6.2.1 Data Collection

For these experiments, I deployed Scrabblesque online and recruited participants via email distribution lists and social networking sites (Facebook, Twitter, etc.). I used snowball sampling as I encouraged participants who had taken the study to share the experiment with their friends and family.

The data collection proceeded in two phases. First, I collected data from 105 players for 195 games. This data was collected without the use of the experience manager and was used to populate the game state database and to construct the target distribution.

During the second phase of data collection, I implemented the DDBIAG algorithm in Scrabblesque and made it publicly available online. By the end of the data collection process, I had obtained 40 games of Scrabblesque. In these games, the computer controlled AI played a total of 297 words. Of those 297 words played, the experience manager selected words according to my algorithm 232 times. This means that the DDBIAG algorithm was used to select 78.1% of all words played by the computer. The reason that the computer does not always play words selected by my algorithm is because it is possible that the algorithm will not find a suitable candidate word to play in the time allotted. This could be due to a poor search strategy (random selection) or due to incomplete coverage of the game database. For whatever reason, if this occurs, then the algorithm reverts to selecting words at random.

For each player turn, I then calculated the deviation-based, vanity analytic values as described in Chapter 4.
6.2.2 Distribution Analysis

To evaluate the effectiveness of my algorithm, I first chose to evaluate how well my algorithm was able to fit my target distribution. You can see a direct comparison of the adaptive distribution (distribution obtained by the DDBIAG algorithm) to the target distribution as well as a comparison between the non-adaptive distribution and the target distribution in Figure 6.1a and Figure 6.1b respectively.

In order to statistically evaluate how well the experience manager fit the target distribution, I calculated the $KL$-divergence between the target distribution and the adaptive distribution. $KL$-divergence is not a true distance, as it is asymmetric; however, it is a well-understood measure with several important properties. In particular, it is consistent, always non-negative, and zero only when the two distributions are exactly equal. In this case, $KL$-divergence measures how well the adaptive distribution approximates the target distribution by measuring the entropy in the target distribution that is unexplained by the adaptive distribution.

To characterize the performance of my algorithm, I also calculated the $KL$-divergence between the target distribution and the non-adaptive distribution obtained during the first phase of data acquisition when the experience manager was not used. Since the $KL$ divergence is an asymmetric divergence value, I chose to turn this into a distance value by using the following formula:

$$\frac{KL(adaptive, target) + KL(target, adaptive)}{2}$$  \hspace{1cm} (6.1)

In the above equation, $KL(adaptive, target)$ indicates that I calculated the $KL$-divergence between the adaptive distribution and the target distribution. As you can see, I calculate the $KL$-divergence in both directions and then take the average to turn it into a distance measure. The results for this analysis can be seen in Table 6.1. As shown in the table, the adaptive version of Scrabblesque outperformed the non-adaptive version when comparing the adaptive distribution against the target distribution; however, the non-adaptive game greatly outperformed my algorithm when comparing the other direction as well as on average. I will return to this result in Section 6.2.5 below.
Figure 6.1 Direct comparisons between the target distribution and the adaptive distribution using my algorithm and non-adaptive distribution. Each graph shows probabilities for each possible feature combination. The adaptive distribution appears to be shifted slightly, but overall matches the target distribution more closely than the non-adaptive distribution.
Table 6.1 KL divergence values. Shows divergence between the adaptive/non-adaptive distributions and the target distribution (Direction 1) and between the target distribution and the adaptive/non-adaptive distributions (Direction 2). The average for each algorithm is also shown.

<table>
<thead>
<tr>
<th></th>
<th>Direction 1</th>
<th>Direction 2</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Adaptive</td>
<td>3.29</td>
<td>3.03</td>
<td>3.14</td>
</tr>
<tr>
<td>Adaptive</td>
<td>2.984</td>
<td>13.86</td>
<td>8.42</td>
</tr>
</tbody>
</table>

Table 6.2 Comparison between the non-adaptive version of Scrabblesque and the adaptive version of Scrabblesque using the DDBIAG algorithm in terms of the number of finished and unfinished games.

<table>
<thead>
<tr>
<th></th>
<th>Finished</th>
<th>Unfinished</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Adaptive</td>
<td>148</td>
<td>47</td>
<td>24.1%</td>
</tr>
<tr>
<td>Adaptive</td>
<td>34</td>
<td>6</td>
<td>15%</td>
</tr>
</tbody>
</table>

6.2.3 Analyzing Quitting Rate

Recall that I authored this target distribution using data collected in previous work. In that work I found that there are certain analytic values that are indicative of players quitting the game early. As a result, I authored this target distribution to minimize the occurrence of those analytic values. Accordingly, one method for evaluating the effectiveness of my algorithm was to see how effective my algorithm was at lowering the frequency of players quitting the game early.

According to the data I had gathered previously on the non-adaptive version of Scrabblesque, 24.1% of all games were ended prematurely. Using my action selection algorithm, I was able to reduce this percentage to 15%. A summary of this result can be seen in Table 6.2. Despite the fact that my algorithm was outperformed by the baseline in fitting the target distribution, a lower percentage of players quit the adaptive version of Scrabblesque than in the non-adaptive version. Using Fisher's exact test, I was unable to confirm that this difference was statistically significant ($p = 0.30$).

6.2.4 Individual Feature Analysis

In order to help explain the KL divergence values that were calculated, I decided to analyze how well the adaptive version of Scrabblesque was able to influence each analytic individ-
Table 6.3 Number of times that players exhibited the desired behavior for each analytic as well as the total number of attempts to elicit this behavior in players.

<table>
<thead>
<tr>
<th></th>
<th>Correct</th>
<th>Total</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Score Difference</td>
<td>168</td>
<td>232</td>
<td>72.4%</td>
</tr>
<tr>
<td>Word Length</td>
<td>127</td>
<td>232</td>
<td>54.7%</td>
</tr>
<tr>
<td>Word Score</td>
<td>184</td>
<td>232</td>
<td>79.3%</td>
</tr>
<tr>
<td>Words Submitted</td>
<td>23</td>
<td>232</td>
<td>9.91%</td>
</tr>
</tbody>
</table>

ually. This was done by examining how often the player’s turn exhibited each individual analytic value that the experience manager targeted. A summary of these values can be seen in Table 6.3.

The most important value to notice is the number of times that players exhibited the desired behavior for the words submitted analytic. Out of 232 attempts, players only displayed the desired behavior 23 times. In other words, my algorithm was only able to effectively influence the player 9.91% of the time with respect to the words submitted analytic.

6.2.5 Marginalized Distribution Analysis

The poor performance of my algorithm on influencing the words submitted analytic caused me to revisit my distribution analysis. In this second analysis, I marginalized the words submitted analytic out of all distributions (the target distribution, the non-adaptive distribution, and the adaptive distribution) and then recalculated the $K L$-divergence using equation 6.1. My hypothesis was that the word submitted feature was not contributing to lowering the percentage of games that ended prematurely. A comparison of the marginalized adaptive distribution to the marginalized target distribution can be seen in Figure 6.2a while a comparison of the marginalized non-adaptive distribution to the marginalized target distribution can be seen in Figure 6.2b. As the figures show, the adaptive distribution seems to fit the target distribution much better than the non-adaptive version.

As you can see in Table 6.4, my intuition was confirmed by the results of the KL divergence analysis. By marginalizing over the words submitted analytic, the performance of my algorithm increases dramatically. In these experiments, the adaptive version of Scrabblesque using the DDBIAG algorithm outperformed the non-adaptive version of the game in all cases, meaning that my algorithm actually is influencing the games that people played.
Figure 6.2 Direct comparisons between the target distribution and the adaptive distribution using my algorithm and the non-adaptive distribution after the word *submitted* analytic has been marginalized out. Each graph shows probabilities for each possible feature combination.
Table 6.4 KL-divergence values after marginalizing the words submitted analytic. Since KL-divergence values differ depending on the order in which a comparison is made, I averaged the results across both directions. Direction 1 refers to the comparison of the adaptive/non-adaptive distribution to the target distribution whereas Direction 2 refers to the comparison of the target distribution to the adaptive/non-adaptive distribution.

<table>
<thead>
<tr>
<th></th>
<th>Direction 1</th>
<th>Direction 2</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Adaptive</td>
<td>0.86</td>
<td>0.51</td>
<td>0.69</td>
</tr>
<tr>
<td>Adaptive</td>
<td>0.22</td>
<td>0.18</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 6.5 Comparison between the non-adaptive version Scrabblesque and the adaptive version of Scrabblesque using the DDBIAG algorithm in terms of the number of finished and unfinished games. Updated to reflect the inclusion of additional data gathered from an additional round of data collection.

<table>
<thead>
<tr>
<th></th>
<th>Finished</th>
<th>Unfinished</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Adaptive</td>
<td>148</td>
<td>47</td>
<td>24.1%</td>
</tr>
<tr>
<td>Adaptive</td>
<td>55</td>
<td>7</td>
<td>11.3%</td>
</tr>
</tbody>
</table>

6.2.6 Additional Data

During the weeks following these initial experiments, data collection continued. Here I wish to report the results of a replication study done with additional games. In total, I have collected 62 games of Scrabblesque.

First, I analyzed the updated quitting rate amongst players. A summary of this data can be seen in Table 6.5. Of the games seen, 55 games were completed whereas 7 games were quit before completion. After performing a chi-squared test, I find that there is a significant difference between this quitting rate and the quitting rate for the non-adaptive version of Scrabblesque ($p = 0.03$, $d.f. = 1$, $\chi^2 = 4.65$). In terms of percentages, we found the quitting rate fell from 24.1% in the non-adaptive version of Scrabblesque to 11.3% in the version of Scrabblesque using the DDBIAG algorithm.

I also compared KL-divergence values using this updated dataset. As you can see in Table 6.6, I found that KL-divergence values for this dataset were 0.59 and 0.36 for the adaptive distribution versus the target distribution and the target distribution versus the adaptive distribution, respectively. This leads to an average KL-divergence value of 0.48, which is still lower than the KL-divergence values achieved by the non-adaptive version of Scrabblesque. This change is reflected in Figure 6.3a. As you can see, my algorithm does
Table 6.6 $KL$-divergence values after marginalizing the words submitted feature. Updated to reflect the inclusion additional data gathered from an additional round of data collection.

<table>
<thead>
<tr>
<th></th>
<th>Direction 1</th>
<th>Direction 2</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-Adaptive</td>
<td>0.86</td>
<td>0.51</td>
<td>0.69</td>
</tr>
<tr>
<td>Adaptive</td>
<td>0.59</td>
<td>0.36</td>
<td>0.48</td>
</tr>
</tbody>
</table>

Table 6.7 The percentage of games that ended in a low, medium, or high number of turns.

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive</td>
<td>24.2%</td>
<td>59.7%</td>
<td>16.1%</td>
</tr>
<tr>
<td>Non-Adaptive</td>
<td>31.4%</td>
<td>38.6%</td>
<td>30.0%</td>
</tr>
</tbody>
</table>

not fit the target distribution as well as it did with less data; however, it still fits the target distribution better than the baseline distribution.

I also chose to analyze the length of games in each treatment. In this analysis, I created bins corresponding to games that ended in a low number of turns, medium number of turns, and high number of turns. Bins were determined by, first, finding the average length of a completed game regardless of treatment. This came out to be 10 turns with a standard deviation of 2 turns. Using these, I defined a low number of turns as less than 8, a medium number of turns as anything between 8 and 12 turns, and a high number of turns as greater than 12 turns. Table 6.7 shows the results of this analysis. As shown in the table, game lengths in the adaptive version of Scrabblesque are skewed towards a medium number of turns whereas the non-adaptive games are a bit more uniform. This shows that games of Scrabblesque using the DDBIAG algorithm are not only more likely to be completed, but they are more likely to contain a medium number of turns.

Finally, I chose to do a surface analysis of how my algorithm affected long-term retention. Although the main purpose of my research is to influence session-level retention, increasing session-level retention at the cost of long-term retention makes it of questionable use to game designers and developers. In this analysis I examined the percentage of players that played more than one game of the non-adaptive version of Scrabblesque as well as the adaptive version. The results of this analysis are shown in Table 6.8.

As you can see, roughly 30% of players in both conditions played multiple games. While this result is not definitive, it provides evidence that the presence of the DDBIAG algorithm does not negatively impact long-term player retention. It also provides evidence that long-
Comparison of Marginalized Target and Adaptive Distributions Replicated

(a) Adaptive and Target Comparison Replicated

Comparison of Marginalized Target and Non-Adaptive Distributions

(b) Non-Adaptive and Target Comparison Replicated

Figure 6.3 Replicated comparison between the target distribution and the adaptive distribution using my algorithm and the non-adaptive distribution after the word submitted analytic has been marginalized out with additional data.

Table 6.8 The percentage of players that played multiple games of Scrabblesque.

<table>
<thead>
<tr>
<th></th>
<th>Adaptive</th>
<th>Non-Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30.8%</td>
<td>30.2%</td>
</tr>
</tbody>
</table>
term retention and session-level retention are independent problems that can be solved using different techniques that will not necessarily negatively impact the other.

6.2.7 Discussion

Perhaps the most interesting finding to come from these experiments is that my algorithm was able to successfully influence player behavior according to the target distribution I supplied in Scrabblesque. This means that I can ensure that players will receive the variety of play experiences I desire since I am generating them according to a distribution of experiences.

Another important result of this work was the promising results with respect to decreasing the percentage of games that ended prematurely. In previous work, I identified features that I determined were indicative of players ending the game early. Since this information was used to author the distribution targeted by my algorithm, I am confident that most of these features are predictive of players quitting the game.

Another interesting affect that the DDBIAG algorithm had on games is that it skewed game length. As shown in Table 6.7, games in the adaptive version of Scrabblesque are more likely to be of a medium length (between 8 and 12 turns) whereas non-adaptive end more uniformly after a low, medium, or high number of turns. It is possible that one side-effect of the DDBIAG algorithm in Scrabblesque is that game lengths tend to cluster around this range. It is not clear if a relationship exists between the length of a game and the probability that a player will quit. This is something that could be explored further in future work.

An unexpected result of these experiments was that associated with the words submitted feature. I determined that this feature does not influence how likely it is for a player to quit the game early. I hypothesize that the reason for this is that this feature can have several different interpretations. Originally, I assumed that this feature could only indicate frustration. I assumed that if a player was submitting several words in a turn, then that could only mean that the player was playing several words that were not being accepted. As it turns out, there was another case that I had not considered. If a player submits a word that then makes other words, they will all be added to the number of words that player has submitted for the turn. Therefore, it is possible that a high number of words submitted could be indicative of a player doing well.
6.3 Analysis in SQ:TG

The success had in influencing player behavior in Scrabblesque does not necessarily mean that I will find the same amount of success in SQ:TG. After all, SQ:TG is a much more complicated environment and the player is given more freedom to act as they please, possibly decreasing the likelihood that they will be able to be influenced by the experience manager. As with Scrabblesque, my experience management system is meant to accomplish two goals in SQ:TG:

- Alter a set of actionable analytics describing the game world in order to induce a target distribution of game states
- Increase session-level retention

In the next sections, I will evaluate how successfully my system accomplishes these two goals.

6.3.1 Data Collection

For these analyses, I performed a data collection lasting 3 weeks. During this data collection, I recruited from popular social media sites, online forums, and mailing lists. I also encouraged people who had played the game to encourage their friends and family to play as well. During this data collection, players were only given the option to play the adaptive version of Sidequest: The Game. In the analyses that follow, I compare the adaptive version of SQ:TG against the non-adaptive version that was used to create the models of player retention (described in Chapter 4). By the end of the data collection, I had gathered 138 game traces.

6.3.2 Distribution Analysis

As with Scrabblesque, I first analyze how well my system is able to induce the targeted distribution of behavior. Figure 6.4 shows a visual comparison of the distribution generated by the non-adaptive version of SQ:TG, the distribution generated by the adaptive version of SQ:TG, and the target distribution. In both Figure 6.4a and Figure 6.4c, it appears as though the adaptive version of SQ:TG is able to better fit the two desired peaks in both Stage 1 and
Figure 6.4 Comparing the target distribution to the distributions created by the adaptive and non-adaptive versions of Sidequest: The Game.
Table 6.9 Jensen-Shannon divergence values comparing the distributions created by the adaptive/non-adaptive version of Sidequest: The Game and the target distribution.

<table>
<thead>
<tr>
<th></th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive</td>
<td>0.12</td>
<td>0.11</td>
<td>0.03</td>
</tr>
<tr>
<td>Non-Adaptive</td>
<td>0.19</td>
<td>0.09</td>
<td>0.09</td>
</tr>
</tbody>
</table>

Stage 3 of the game. It is more difficult to determine which version of the game better fits the target distribution for Stage 2 of the game (seen in Figure 6.4b).

To statistically evaluate these distributions, I calculate the Jensen-Shannon divergence (Lin, 1991) between each distribution and the target distribution. Jensen-Shannon divergence is considered to be a more general form of KL-divergence. It is used here because using KL-divergence requires that the distributions be absolutely continuous with respect to each other. This means that no part in either distribution being tested can occur with 0 probability. In Scrabblesque, the distributions created met this requirement; however, the distributions produced by SQ:TG do not. Jensen-Shannon divergence relaxes this requirement, making it applicable to the distributions in SQ:TG. Another added benefit of Jensen-Shannon divergence is that it is a symmetric measure, meaning that calculations do not need to be made in both directions and then averaged.

Results of this analysis are shown in Table 6.9. As you can see, the adaptive version of SQ:TG is able to produce a distribution that better fits the target distribution in 2 out of the 3 stages of the game. Stage 2 was the only stage where I did not see a decrease in Jensen-Shannon divergence.

6.3.3 Quitting Rate Analysis

The second analysis performed is a comparison of session level retention as measured by the quitting rate in the adaptive and non-adaptive versions of SQ:TG. A summary of this data is shown in Table 6.10.

As seen in the table, the quitting rate in the adaptive version of SQ:TG is 34.10% while the quitting rate in the non-adaptive version of SQ:TG is 47.00%. This is a difference of 12.9%. I used Fisher’s exact test to measure to determine if this difference was statistically significant. Using this test I found that, with a p-value of 0.015, this difference is statistically significant.

In addition to this general quitting rate analysis, I have performed an analysis on when
Table 6.10 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.

<table>
<thead>
<tr>
<th></th>
<th>Finished</th>
<th>Unfinished</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive</td>
<td>91</td>
<td>47</td>
<td>34.1%</td>
</tr>
<tr>
<td>Non-Adaptive</td>
<td>141</td>
<td>125</td>
<td>47.0%</td>
</tr>
</tbody>
</table>

Table 6.11 The percentage of players that completed each stage of *Sidequest: The Game* in the adaptive and non-adaptive versions.

<table>
<thead>
<tr>
<th></th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive</td>
<td>72.5%</td>
<td>65.9%</td>
<td>65.9%</td>
</tr>
<tr>
<td>Non-Adaptive</td>
<td>64.7%</td>
<td>57.5%</td>
<td>53.8%</td>
</tr>
</tbody>
</table>

players quit. In this analysis, I examined how many players completed each stage of the game to determine at which stage of the game players are quitting. These results are shown in Table 6.11.

These results show that the adaptive version was able to better retain players at each stage of SQ:TG. For both Stage 1 and Stage 2, the difference in completion percentages is roughly 8%. The difference in completion percentages for Stage 3 grows to roughly 12% since everyone who completed Stage 2 in the adaptive version of SQ:TG also completed Stage 3. It is also interesting to view these results with respect to how many people were lost at each stage. This data is shown in Table 6.12. While it is true that the adaptive version of SQ:TG loses fewer people at each stage, it is interesting to note that during Stage 2 the percentage of players lost is similar to the amount of players lost in the non-adaptive version. This will be discussed in greater detail in the next section.

Table 6.12 The percentage of players that quit at each stage of *Sidequest: The Game*

<table>
<thead>
<tr>
<th></th>
<th>Stage 1</th>
<th>Stage 2</th>
<th>Stage 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive</td>
<td>27.5%</td>
<td>6.6%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Non-Adaptive</td>
<td>35.3%</td>
<td>7.2%</td>
<td>3.7%</td>
</tr>
</tbody>
</table>
Table 6.13 The percentage of players that quit at each stage of *Sidequest: The Game*

<table>
<thead>
<tr>
<th></th>
<th>Adaptive</th>
<th>Non-Adaptive</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0%</td>
<td>1.2%</td>
</tr>
</tbody>
</table>

### 6.3.4 Long-Term Retention Analysis

In *Scrabblesque*, I recorded how many people played multiple games and found that about 30% of players played multiple games in both versions of *Scrabblesque*. A similar analysis was performed on the data collected from SQ:TG. Table 6.13 shows that SQ:TG was not able to get many players to play multiple games in either condition. This is likely due to the length of the game. A single game of *Scrabblesque* can be completed in about 5 minutes whereas it can take upwards of 30 minutes to finish SQ:TG. This might discourage players from wanting to invest the time in multiple playthroughs. Regardless, this result also shows that the DDBIAG algorithm did not seem to definitively affect long-term retention in either a positive or negative way.

### 6.3.5 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 me to believe that this technique is fully capable of affecting player behavior in this environment under the right circumstances.

In stage 2, however, both versions of the game performed similarly (with the non-adaptive version performing slightly better) with respect to fitting the target distribution. One explanation for this phenomenon has to do with the complexity of the distribution that needed to be fit and the relative complexity of the game environment. In stage 2 of SQ:TG, ideal behavior involves some amount of exploration through the space of possible quest interactions. This manifests as accepting some quests and then rejecting others. The system does this by, first, offering the player both quests that they are likely to accept and quests they are likely to reject. This is because there are often more than one quest-giver in a location, which means that the system needs to make it possible for the player to both accept and reject a quest in the same location. It is possible that this mixing of goals caused the adaptive version of SQ:TG to perform similarly to the non-adaptive version in this case.
The quitting rates observed are as expected given how well the adaptive version of the game was able to fit the target distribution. I found that my technique was able to produce a statistically significant drop in the quitting rate, just as it did in Scrabblesque. A more detailed examination of when people were quitting showed that the adaptive version of SQ:TG saw a higher percentage of players complete each act than the non-adaptive version. Examining the amount of players lost moving from act to act yielded interesting results as well. This showed that the largest differences in players lost between the adaptive and non-adaptive versions of the game occurred at the very beginning (moving from the start of the game to the end of Stage 1) and at the very end (moving from the start of Stage 3 to the end of the game). This behavior is not surprising given the performance of my technique on fitting the target distribution of Stage 2. It is understandable that there is a similar drop in the number of players given that both versions of the game behaved similarly at that stage of the game.

It is also interesting to note that the target distribution of player behaviors was much simpler in SQ:TG than it was in Scrabblesque. The fact that the DDBIAG algorithm was able to indirectly influence player behavior and increase session-level retention in both of these environments gives evidence to its generalizability on many levels. Not only was it proven to be effective in a relatively simple game environment (Scrabblesque) as well as a more complex game environment (SQ:TG), but it was able to fit both a complex target distribution (Scrabblesque) as well as a relatively uncomplicated target distribution (SQ:TG).

The most important takeaway from these experiments is that my technique was able to accomplish both of the goals that I set out to do, namely influencing player behavior to fit a targeted distribution of behavior and reducing the quitting rate in SQ:TG. This, at the very least, provides evidence that a relationship exists between the shifts in player behavior that my system is able to produce and the subsequent reductions in quitting rate.

6.4 Conclusion

After evaluating the effectiveness of the DDBIAG algorithm in both Scrabblesque and Side-quest: The Game, I can say that this algorithm was able to effectively influence behavior and lower the quitting rate in both game environments. This shows that not only is DDBIAG effective at influencing quitting rate, but it also shows that it is effective at bridging the gap between actionable and vanity analytics since it is able to incorporate both of them
into an experience management scheme. These experiments show that the DDBIAG algorithm is able to indirectly influence, through the manipulation of actionable analytics, player behavior in both simple and complex casual game environments. Also, the DDBIAG algorithm is able to induce relatively complex distributions of behavior as well as relatively simple distributions of behavior in these environments.
CHAPTER 7

PSYCHOMETRIC SIDE-EFFECTS

7.1 Introduction

While the main goal of this research is to increase session-level retention in games, it is possible that the DDBIAG algorithm could have side-effects on behavior beyond player-retention. In particular, I am interested in the effects that the DDBIAG algorithm had on player engagement and intrinsic motivation. The reason that I am concerned with these metrics is because they are two common measures of player experience and it is feasible that my technique could have had an effect on them. In this chapter, I discuss a set of user studies that I ran in both Scrabblesque and Sidequest: The Game in order to measure player engagement and intrinsic motivation in both the adaptive versions of these games as well as the non-adaptive versions. I will also discuss, in detail, the instruments used to perform these studies. For a more detailed discussion of engagement and intrinsic motivation, I direct the reader to the discussion in Chapter 2.
7.2 Psychometric Instruments

In my experiments on intrinsic motivation and engagement, I use two validated instruments for measuring these psychometric phenomenon: the intrinsic motivation inventory (IMI) and the games engagement questionnaire (GEQ). I will discuss both of these instruments in greater detail below.

7.2.1 Intrinsic Motivation Inventory

In order to measure intrinsic motivation, I am using the Intrinsic Motivation Inventory (IMI). The IMI is a multidimensional measurement device used to determine a participant’s intrinsic motivation. Use of the full measurement device will yield seven subscores that refer to the following:

- Interest/Enjoyment
- Perceived Competence
- Effort
- Value/Usefulness
- Felt Pressure and Tension
- Perceived Choice
- Relatedness

Currently, only the Interest/Enjoyment subscale is considered to be a self-report measure of intrinsic motivation (McAuley et al., 1989). As a result, in the experiments that I will perform I will only use questions drawn from the “Interest/Enjoyment” subscale.

The “Interest/Enjoyment” subscale consists of 7 statements including two statements that are reversed. A reversed statement is a statement that measures the opposite of what the subscale is attempting to measure. These reversed questions are meant to add validity to the results gathered using this measure by controlling for people that may not be paying attention while responding to these items. As previously stated, this subscale consists of 7 statements. Participants are given these statements and then asked to rate their agreement with these statements on a scale of 1 to 7 with 1 corresponding to “Not True At All”, 4
Table 7.1 The Intrinsic Motivation Inventory. Responses are measured on a 7-point Likert scale with 1 meaning *not true* and 7 meaning *very true*.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>This game was fun.</td>
</tr>
<tr>
<td>2.</td>
<td>This game did not hold my attention at all.</td>
</tr>
<tr>
<td>3.</td>
<td>I thought this game was quite enjoyable.</td>
</tr>
<tr>
<td>4.</td>
<td>I enjoyed playing this game very much.</td>
</tr>
<tr>
<td>5.</td>
<td>I thought this was a boring game.</td>
</tr>
<tr>
<td>6.</td>
<td>I would describe this game as very interesting.</td>
</tr>
<tr>
<td>7.</td>
<td>While I was playing this game, I was thinking about how much I enjoyed it.</td>
</tr>
</tbody>
</table>

corresponding to “Somewhat True”, and 7 corresponding to “Very True”. The statements used in this subscale can be seen in Table 7.1. In this table, statements 2 and 5 are the reversed statements.

### 7.2.2 Game Engagement Questionnaire

The Game Engagement Questionnaire (GEQ) (Brockmyer et al., 2009) is a measurement tool used to measure the levels of engagement experienced while playing video games. The GEQ consists of 19 statements that are meant to measure several aspects of engagement. Examples of the aspects of engagement that are measured include flow, immersion, and presence. Participants report their agreement with these statements using a 3 point scale with 1 indicating disagreement, 2 indicating that the participant is unsure of agreement or disagreement, and 3 indicating agreement.

The statements present in the GEQ are meant to measure 4 different facets of engagement: immersion, presence, flow, and absorption. In this questionnaire, 1 statement measures immersion, 4 statements measure presence, 9 statements measure flow, and 5 statements measure absorption. For my experiments, I have chosen to use all statements present in the GEQ. Table 7.2 lists the statements on the GEQ that were to my participants. In this table, statement 1 is meant to measure immersion, statements 2 - 5 are meant to measure presence, statements 6 - 14 are meant to measure flow, and statements 15 - 19 are meant to measure absorption. It is important to note that the order that these questions were given to participants has been randomized to remove any bias that grouping the questions based on what they are measuring may have introduced.
**Table 7.2** The Intrinsic Motivation Inventory. Responses are measured using a 3-point Likert scale with 1 meaning disagreement, 2 meaning neither agreement or disagreement, and 3 meaning agreement.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>I really get into the game.</td>
</tr>
<tr>
<td>2.</td>
<td>I lose track of time.</td>
</tr>
<tr>
<td>3.</td>
<td>Things seem to happen automatically.</td>
</tr>
<tr>
<td>4.</td>
<td>My thoughts go fast.</td>
</tr>
<tr>
<td>5.</td>
<td>I play longer than I mean to.</td>
</tr>
<tr>
<td>6.</td>
<td>The game feels real.</td>
</tr>
<tr>
<td>7.</td>
<td>If someone talks to me, I don't hear them.</td>
</tr>
<tr>
<td>8.</td>
<td>I don't answer when someone talks to me.</td>
</tr>
<tr>
<td>9.</td>
<td>I can't tell that I’m getting tired.</td>
</tr>
<tr>
<td>11.</td>
<td>I play without thinking how to play.</td>
</tr>
<tr>
<td>12.</td>
<td>Playing makes me feel calm.</td>
</tr>
<tr>
<td>13.</td>
<td>I feel like I just can't stop playing.</td>
</tr>
<tr>
<td>15.</td>
<td>I feel different.</td>
</tr>
<tr>
<td>16.</td>
<td>I feel scared.</td>
</tr>
<tr>
<td>17.</td>
<td>Time seems to kind of stand still or stop.</td>
</tr>
<tr>
<td>18.</td>
<td>I feel spaced out.</td>
</tr>
<tr>
<td>19.</td>
<td>I lose track of where I am.</td>
</tr>
</tbody>
</table>
Table 7.3 Summary of IMI results in Scrabblesque

<table>
<thead>
<tr>
<th></th>
<th>Participants</th>
<th>Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Game</td>
<td>20</td>
<td>37.47 ± 6.70</td>
</tr>
<tr>
<td>Non-Adaptive Game</td>
<td>19</td>
<td>32.63 ± 4.83</td>
</tr>
</tbody>
</table>

Table 7.4 Summary of GEQ results in Scrabblesque

<table>
<thead>
<tr>
<th></th>
<th>Participants</th>
<th>Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Game</td>
<td>19</td>
<td>30.31 ± 4.30</td>
</tr>
<tr>
<td>Non-Adaptive Game</td>
<td>18</td>
<td>26.61 ± 5.40</td>
</tr>
</tbody>
</table>

7.3 Psychometric Evaluation in Scrabblesque

The first study run was to determine if there were any side-effects on intrinsic motivation and player engagement in Scrabblesque. The following sections discuss the details and results of this user study.

7.3.1 Methodology and Data Collection

To measure the possible side-effects that the DDBIAG had on intrinsic motivation and engagement in Scrabblesque, I performed a study in which participants were asked to play 1 of 2 possible versions of the game at random. Upon the completion of their first game, they were asked if they would like to participate in a short survey. If they declined, they were given the choice to play another game or not and then never asked to take the survey again. If the player chose to take the survey, they were then presented with the surveys described above. The order of survey statements was randomized; however, the IMI was always given before the GEQ. Participants were not required to answer any of the questions and could move on to the next part of the survey at any time. It is important to note that this data collection was done separately than the previous data collections done for model building and dynamic game adaption in Scrabblesque.

I recruited participants over the course of 3 weeks from a combination of mailing lists and social media web sites. I also encouraged participants to direct their friends and family to this study in order to gather more participants.

At the end of this 3 week period, 47 participants had chosen to take the survey. Since
Table 7.5 T-Test Results for the comparison of the non-adaptive version of Scrabblesque against the version of Scrabblesque with the DDBIAG algorithm. Results are given for both the IMI and the GEQ.

<table>
<thead>
<tr>
<th></th>
<th>P Value</th>
<th>T value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intrinsic Motivation</td>
<td>0.01</td>
<td>2.55</td>
</tr>
<tr>
<td>Engagement</td>
<td>0.01</td>
<td>2.30</td>
</tr>
</tbody>
</table>

participants could skip portions of the survey, I chose to only look at those participants that completed the survey in full. After doing this data cleaning, I had 39 people who had completed the IMI and 37 people who had completed the GEQ. A summary of the data can be found in Table 7.3 and in Table 7.4.

7.3.2 Analysis of Intrinsic Motivation and Engagement

As you can see in Table 7.3 and in Table 7.4, the adaptive version of Scrabblesque outperforms in terms of average score for both intrinsic motivation and engagement. To verify that these differences were statistically significant, I ran a two-tailed independent samples t-test. It is important to note that I am using parametric statistical tests and reporting the means and standard deviations of this data because I am working with scores rather than categorical data. Since these two surveys have given the accepted way to score each of them, I can treat these scores as numerical data rather than categorical data.

As you can see in Table 7.5, the differences in response scores between the adaptive version of Scrabblesque and the non-adaptive version of Scrabblesque are statistically significant for both the IMI ($p = 0.01$) and the GEQ ($p = 0.01$). This difference implies that the presence of my experience manager in Scrabblesque had a statistically relevant effect on both the intrinsic motivation experienced by people who played the game and the amount of engagement that those players felt.

7.3.3 Analysis of the Aspects of Engagement

Since the GEQ contains 4 subscales that each measure intensities of engagement, I examined how the presence of my experience manager affected each of these subscales. To perform this experiment, I performed a one-tailed independent samples t-test on each subscale of the GEQ. The results of this analysis can be seen in Table 7.6. According to the table, I did not observe a statistically significant difference in response values for shallow forms...
Table 7.6 Summary of data and T-Test Results for GEQ Subscales in Scrabblesque

<table>
<thead>
<tr>
<th></th>
<th>Non-Adaptive Average</th>
<th>Adaptive Average</th>
<th>P-Value</th>
<th>T Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immersion</td>
<td>1.58 ± 0.77</td>
<td>1.90 ± 0.91</td>
<td>0.12</td>
<td>1.19</td>
</tr>
<tr>
<td>Presence</td>
<td>9.74 ± 1.52</td>
<td>9.75 ± 1.55</td>
<td>0.49</td>
<td>0.03</td>
</tr>
<tr>
<td>Flow</td>
<td>12.89 ± 3.11</td>
<td>14.45 ± 3.44</td>
<td>0.07</td>
<td>1.48</td>
</tr>
<tr>
<td>Absorption</td>
<td>8.56 ± 1.85</td>
<td>9.89 ± 2.53</td>
<td>0.04</td>
<td>1.84</td>
</tr>
</tbody>
</table>

of engagement ($p = 0.12$ and $p = 0.49$ for immersion and presence respectively); however, I observed a marginally significant difference ($p = 0.07$) in flow response values and a statistically significant difference ($p = 0.04$) in absorption response values.

### 7.4 Discussion

My qualitative analysis showed that the games produced by my experience manager are more intrinsically motivating and engaging than the games produced by the non-adaptive version of Scrabblesque. This means that my experience manager can influence game states in a variety of ways and that this brings about an increase in player intrinsic motivation and engagement.

The results of the aspects of engagement analysis merit further discussion as well. The results show that my adaptive version of Scrabblesque performed comparably to the non-adaptive version with respect to immersion and presence. According to (Brockmyer et al., 2009), this indicates that the non-adaptive game and the adaptive perform similarly for measures of shallow engagement. I feel that this is expected since, as Brockmyer et al. explain, feelings of immersion and presence are not uncommon in games. Notice that as I move to measures of deeper engagement, such as flow and absorption, the adaptive version of Scrabblesque begins to outperform the non-adaptive version of the game. In other words, the adaptive version of Scrabblesque enables players to feel a deeper sense of engagement whereas both versions enable players to feel a shallow sense of engagement.

### 7.5 Psychometric Evaluation in Sidequest: The Game

My experiments in Scrabblesque showed that one of the side-effects of the DDBIAG algorithm in this environment was that player engagement and intrinsic motivation increased.
Table 7.7 Summary of IMI results in *Sidequest: The Game*

<table>
<thead>
<tr>
<th></th>
<th>Participants</th>
<th>Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Game</td>
<td>84</td>
<td>28.60 ± 10.26</td>
</tr>
<tr>
<td>Non-Adaptive Game</td>
<td>134</td>
<td>23.99 ± 8.74</td>
</tr>
</tbody>
</table>

Table 7.8 Summary of GEQ results in *Sidequest: The Game*

<table>
<thead>
<tr>
<th></th>
<th>Participants</th>
<th>Average Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Game</td>
<td>88</td>
<td>28.65 ± 6.83</td>
</tr>
<tr>
<td>Non-Adaptive Game</td>
<td>122</td>
<td>24.93 ± 4.47</td>
</tr>
</tbody>
</table>

This set of experiments explores if the DDBIAG algorithm has a similar set of side-effects in SQ:TG.

### 7.5.1 Methodology and Data Collection

During the data collections for SQ:TG that were detailed in Chapter 4 (for model building) and in Chapter 6 (for testing the adaptive version of SQ:TG), players were offered the chance to take the IMI and the GEQ once they had completed the game. These surveys were optional and players were not required to answer every question on either survey. As with the experiment performed on *Scrabblesque*, players always took the IMI before the GEQ, but the order of the individual questions on each survey was randomized.

After all data collections were completed, 134 players had completed the IMI and 122 players had completed the GEQ in the non-adaptive version of SQ:TG. In the adaptive version of SQ:TG, 84 people finished the IMI and 88 people completed the GEQ. Table 7.7 contains a summary of the IMI survey data, and Table 7.8 contains a summary of the GEQ survey data.

Recall that, as with *Scrabblesque*, survey responses for the GEQ and IMI were only considered if the player had responded to every statement. So, if a player responded to each statement on the GEQ but not on the IMI, then their responses would be used in the analysis on the GEQ, but not on the IMI.
### 7.5.2 Analysis of Intrinsic Motivation and Engagement

Examining Table 7.7 and Table 7.8 show that the adaptive version of SQ:TG outperforms the non-adaptive version for both player engagement and intrinsic motivation. On the IMI, the players that played the adaptive version of SQ:TG reported a score of 28.60 on average while players who played the non-adaptive version reported an average score of 23.99. For the GEQ, players who played the adaptive version of the game reported an average score of 28.65 while players who played the non-adaptive version reported an average score of 24.93. To test whether these differences were statistically significant, I used a two-tailed, independent samples T-test. This test found that the differences were statistically significant for both the IMI ($p < 0.05$) and the GEQ ($p < 0.05$). These results are summarized in Table 7.9. This means that one of the side affects of the DDBIAG algorithm is the ability to increase player engagement and intrinsic motivation in this game environment.

### 7.5.3 Analysis of the Aspects of Engagement

Recall that the GEQ is comprised of questions that are meant to measure 4 different aspects of engagement: immersion, presence, flow, and absorption. Also recall that these four aspects of engagement have a hierarchical relationship with immersion being the shallowest form of engagement and absorption being the deepest form of engagement. In this analysis, I look at how the presence of the DDBIAG algorithm in SQ:TG affected each of these subscales. The results of this analysis are shown in Table 7.10.

As seen in the table, the adaptive version of SQ:TG produces higher scores for each subscale of the GEQ, although these differences are not always statistically significant. For the shallower forms of engagement (immersion and presence), the differences reported were not statistically significant ($p = 0.32$ and $p = 0.25$ for immersion and presence respectively). Moving to the deeper forms of engagement, however, I found that the differences between the non-adaptive version of SQ:TG and the adaptive version become statistically significant.
Table 7.10 Summary of data and T-Test Results for GEQ Subscales in Sidequest: The Game

<table>
<thead>
<tr>
<th></th>
<th>Non-Adaptive Average</th>
<th>Adaptive Average</th>
<th>P-Value</th>
<th>T Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Immersion</td>
<td>1.51 ± 0.67</td>
<td>1.60 ± 0.70</td>
<td>0.32</td>
<td>1.00</td>
</tr>
<tr>
<td>Presence</td>
<td>6.35 ± 1.80</td>
<td>6.64 ± 1.95</td>
<td>0.25</td>
<td>1.16</td>
</tr>
<tr>
<td>Flow</td>
<td>12.64 ± 2.46</td>
<td>14.86 ± 3.86</td>
<td>8.67e−7</td>
<td>5.07</td>
</tr>
<tr>
<td>Absorption</td>
<td>6.08 ± 1.44</td>
<td>6.89 ± 2.31</td>
<td>0.002</td>
<td>3.21</td>
</tr>
</tbody>
</table>

(p = 8.67e−7 and p = 0.002 for flow and absorption respectively).

7.5.4 Discussion

Perhaps the most interesting result of this analysis is that the presence of the DDBIAG algorithm in SQ:TG was able to produce higher intrinsic motivation and engagement in players as a side-effect. One thing to note, however, is that IMI and GEQ values were consistently lower in SQ:TG than it is in Scrabblesque. So while I find that there are similar gains in engagement/intrinsic motivation across game environments, SQ:TG does not seem to be as engaging/intrinsically motivating as Scrabblesque. I attribute this to the underlying design of each game environment. Scrabblesque is simply a Flash implementation of Scrabble, a game that has been around since 1938. This game has managed to survive the test of time, meaning that it is not surprising to find that it has high engagement and intrinsic motivation values associated with it. SQ:TG, however, is a game of my own creation. Since I am not an expert in game design by any means, it is likely that SQ:TG simply does not provide as engaging or intrinsically motivating an experience as Scrabblesque. That being said, the DDBIAG was still able to increase these values in SQ:TG despite the lower initial values.

The analysis of the aspects of engagement give results similar to those found in Scrabblesque. For shallower forms of engagement, there are no significant differences between the adaptive and non-adaptive versions of the game; however, for deeper forms of engagement the differences become significant. In other words, the version of the game using the DDBIAG algorithm is able to produce a deeper form of engagement in players while still maintaining the shallower forms of engagement. As with Scrabblesque, this is not an unexpected result since most games are able to produce shallow forms of engagement.
7.6 Conclusion

The results of these studies show that the DDBIAG algorithm has a statistically significant effect on both intrinsic motivation and player engagement in both game environments. Results also indicate that another side-effect of the DDBIAG algorithm was that players experienced deeper forms of engagement with the adaptive game environments than with their non-adaptive counterparts. While the main goal of this algorithm was to improve session-level retention, it is promising that it was able to do so without sacrificing player experience. Even though this was not intended, the DDBIAG algorithm could also possibly be used to target game states with the specific goal of improving these types of psychometrics. This avenue of research, however, is left for future work.
The primary contributions of this work are an algorithm for implementing dynamic game adaptations to increase session-level retention and an empirical evaluation of said algorithm in two game environments. This algorithm incorporates both vanity and actionable analytics to determine how to adapt a game environment to increase session-level retention. In doing so, the algorithm bridges the gap between vanity and actionable analytics such that the predictive power of vanity analytics can be used to create changes to game environments. This will hopefully be of use to both game designers and game researchers as it provides a way to use translate between the vanity analytics that are used to create models of player behavior and the actionable analytics used to create changes in game environments.

8.1 Summary of Work

Recall that my thesis statement is as follows:

**Thesis** Data-driven techniques can be used to bridge the gap between the predictive power of vanity analytics and the prescriptive power of actionable analytics to dynamically
adapt game experiences in order to influence player behaviors such as session-level retention in a variety of game environments.

From this thesis statement, the two main phases of my work arose which made up my primary contributions:

- The creation of a data-driven backwards induction algorithm to bridge the gap between vanity and actionable analytics to generate game adaptions in real time
- A quantitative evaluation of the models of session-level retention as well as the DDIAG algorithm

In addition to these contributions, I also performed a psychometric evaluation in each game environment to outline the side-effects that the DDBIAG algorithm had on player experience.

The first phase of my work was to show how vanity analytics could be used to model session-level retention in two game environments: Scrabblesque and Sidequest: The Game. This is done by using an $n$-gram model to find sequences of actions (in terms of vanity analytics) that are highly predictive of players quitting the game early. These sequences are then used to create a target distribution of game states in which those action sequences are avoided. In other words, this target distribution is designed to avoid states that are known to be associated with low session-level retention.

The DDBIAG algorithm then attempts to realize this target distribution by iteratively selecting intermediate goal states from this distribution and making changes to the game world that will move the current game state closest to the goal state. The DDBIAG algorithm determines how to alter the world by finding the set of previously observed game states that match the goal state and expressing them in terms of a set of actionable analytics. It then alters the actionable analytics that describe the current game state so that those analytics closely resemble the actionable analytics displayed by the set of goal states.

My quantitative evaluation of the DDBIAG algorithm showed that when it was used to create dynamic game adaptions, a higher percentage of players finished games when compared to players that played non-adaptive versions of the same games. In addition, an analysis of player behavior distributions in both game environments showed that the players in the adaptive game environments exhibited behaviors that better fit the target distribution of behavior than players in the non-adaptive game environments. This provides...
strong evidence that the DDBIAG algorithm is effective at both influencing player behavior and increasing session-level retention in these game environments. Given the disparity of genre and game mechanics between the two game environments tested, this also provides evidence that the DDBIAG algorithm could generalize to other games as well.

The final evaluation I performed looked to see what effect, if any, the DDBIAG algorithm had on two psychometrics measuring player experience: intrinsic motivation and engagement. The results of this study showed that the DDBIAG algorithm actually improved both intrinsic motivation and player engagement in both Scrabblesque and SQ:TG. In addition, the study showed that players that played the adaptive version of both games experienced deeper forms of engagement when compared to the players that played the non-adaptive version. This gives evidence that the DDBIAG algorithm does not negatively affect play experience and suggests that its presence might lead to more engaging and intrinsically motivating experiences.

8.2 Limitations

One of the greatest strengths of this work also happens to be one of its greatest weaknesses. The fact that the DDBIAG algorithm is data-driven means that it requires a nontrivial amount of player observations before it can be used. Since models must first be built on a non-adaptive version of the game, some amount of time must be spent to gather a sufficient number observations to make these models in a game environment that, in all likelihood, will not be available to the public yet. Depending on the complexity of the game environment and the complexity of the models that will be made, this could be a difficult task since it will take time to collect these observations. Another limitation related to the fact that this is a data-driven technique is that it can be difficult to incorporate new data into the models that are created once people are playing the adaptive version of the game. Since the adaptive version of the game attempts to alter the way that users would normally play, using these to strengthen the models of player behavior used would actually introduce a bias towards the behaviors that the adaptive system is producing. This means that new data can only be used if it comes from players that are playing a non-adaptive version of the game, which could be difficult to obtain once the adaptive version has been made available.

The results of the analysis of player behavior distributions in SQ:TG also allude to another possible limitation of the algorithm. In this analysis, the algorithm failed to influence
player behavior when it tried to induce a relatively complicated behavior distribution in that environment. It is possible that the complexity of this game environment was a contributor in this behavior. I drew this conclusion because I did not observe similar behavior in *Scrabblesque*. If this is the case, then that means that in more complex game environments the DDBIAG algorithm would need to target relatively simple target behavior distributions in order to adapt player behavior.

Another limitation of this algorithm is that it does require some amount of authorial input in the selection of the vanity analytics and actionable analytics involved in model creation and game adaption. It is possible that those with little to no experience selecting analytics for modeling will find this difficult. Since much of the success of the algorithms depend on which analytics are used for modeling and adaption, careful selection of these analytics is completely necessary. One avenue of future work involves attempting to automate this process so that expertise is no longer required to select sets of analytics for the DDBIAG algorithm.

It is also not entirely clear how this technique could be applied to larger games where defining a game session becomes problematic. In most casual games, it usually easy to define when the natural beginning and ending of a game session are. If I move beyond the scope of casual games, it becomes more difficult to determine when a session should begin and when a session should end. It is possible that game sessions could be defined in terms of the objectives that players need to complete. Simply defining possible places for a game session to end to coincide the the player completing some objective or set of objectives; however, it is not clear that this would be the same as a game session in casual games which are often already divided up into levels or something similar.

### 8.3 Impact

For casual game designers, the ability to dynamically increase session-level player retention has many financial benefits. Since the revenue generated by most casual games comes from advertisement traffic in-game or from microtransactions, increasing session-level retention means that players have more opportunities to spend money on these types of services. The ability to influence session-level retention is also useful for serious games or other types of learning environments. The ability to keep students engaged with the learning environment is a base requirement for learning in these environments. If the student stops
playing the game before the lesson has ended or is not engaged during the experience, then it is difficult for the game environment to encourage learning.

For both game researchers, the ability to bridge the gap between vanity and actionable analytics is useful because it means that it is possible, using this technique, to use models of player behavior to create dynamic game environments. To this point, much of the work in this area has used expert knowledge or other knowledge engineering methods to create adaptive game experiences. Using this technique, it is possible to do so in a data-driven way.

It is also possible that this technique is not limited to session-level retention. While the analysis in this dissertation is limited to session-level retention, there is nothing in the DDBIAG algorithm that is unique to session-level retention save the models of player behavior. If the DDBIAG algorithm can be used to induce different types of behavior, then this means that it becomes possible to quickly build adaptive game environments or augment currently existing game environments such that they are capable of producing a wide variety of player experiences while still eliciting a smaller set of desired player behaviors.

In theory, this technique could be useful to anyone making an adaptive system, even if these system are not game environments. One simply needs to have a way to describe the environment involved as a set of vanity and actionable analytics. This could find uses in adaptive/intelligent user interface design or adaptive hypermedia.

8.4 Future Work

This research presents many possible for continuing work. One of these is to explore the space of behaviors that this technique can be applied to. In this dissertation, I singled out a very specific behavior to model (session-level retention) and did not attempt to make claims about any other behaviors. That being said, there is nothing in the DDBIAG algorithm or the techniques used for player modeling that leads me to believe that they exclusively apply to session-level retention. In theory, a variety of different behaviors could be induced using these techniques and it would be interesting to test the limits of what this algorithm is capable of in that respect.

Similarly, this dissertation only examined how the DDBIAG algorithm performed in two game genres. While its performance suggest that it should generalize to other game genres, explicitly testing in which genres the DDBIAG algorithm performs well would provide useful
insights. Along this same line of research, exploring how this technique performs outside of casual games could be done. As I mentioned previously, one of the possible limitations of this algorithm is that it may be difficult to apply to games that do not have a clear session beginning and session end. By moving outside the realm of casual games one can test this hypothesis directly. Since very few “traditional” video games have a clear cut session beginning and session end, testing the DDBIAG algorithm's ability to perform well in these types of games will show if having a clear session beginning and end is necessary to use this algorithm.

Also mentioned earlier is the idea that there is some amount of expert knowledge required to use this technique since a set of vanity and actionable analytics needs to be created for the DDBIAG algorithm to work. If this process could be automated, then there is almost no authorial input needed to use the DDBIAG algorithm, which would make it very appealing for game designers with very little AI or computational modeling expertise. This work could also be useful outside of the DDBIAG algorithm in that it could be used to determine what aspects of a game environment would be the most useful for modeling and influencing various types of behavior. This means that there would be very little prior knowledge required to incorporate computational modeling and analytics into the game design process.

Finally, I suggested earlier that it is possible that this technique could be used for other types of adaptive systems. In particular I would be interested in how this technique could be used to generate dynamic experiences in serious games or other types of learning environments. In these types of environments, the ability to keep a user engaged until the end of a lesson is crucial in many of these systems. The ability to improve session-level retention in these types of environments could be very beneficial.

8.5 Concluding Remarks

It has long been a goal of both game designers and game researchers to understand player retention in games. Designers often rely on expert knowledge to determine how to best increase player retention in game environments. These techniques are often lack rigorous validation which means that the accuracy of these rules can be suspect. Researchers, on the other hand, have focused their efforts on building computational models of player retention. These models, however, often do not clearly indicate how their insights can
be used in a game environment since they are often constructed using vanity analytics. This work improves upon both of these methods for player retention though the use of a data-driven technique for dynamic game adaption that bridges the gap between vanity and actionable analytics. This approach improves upon the knowledge engineering approaches used by game designers by grounding all insights in data. This means that all actions taken by my technique are done because observed behaviors indicated that these actions should be taken. By bridging the gap between vanity and actionable analytics, the types of models often created by researchers can be incorporated into game environments to improve player retention.

This work is a significant step forward in the area of retention as it provides evidence of actually increasing player retention (albeit at the session level) in multiple game environments. This work also serves to show the power that data-driven techniques for behavior modeling and action generation have in game environments. Moving forward, I hope that the insights contained in this dissertation inspire future researchers to explore the use of these techniques and how they can be applied not only to game environments, but to other, more general adaptive systems.


Nguyen ThaiSon and Lukas Siemon. Impact of sequence mining on web-page recommendations in an access-log-driven recommender system.


As mentioned in Chapter 2, knowledge modeling is concerned with predicting a student's academic ability based on their performance while using an automated tutoring system. In this field, three techniques for performing knowledge modeling have emerged and proven time and time again to be effective in modeling student knowledge. In the next sections, I will go over these techniques in detail and provide examples of systems that incorporate these models.

A.1 Knowledge Tracing

The most commonly used technique for student modeling is knowledge tracing. Knowledge tracing involves monitoring a student's performance on various questions and then uses their performance on these questions to update a model that describes their mastery of the skills, or knowledge concepts (KCs), required to answer the questions.
Figure A.1 A DBN representing student knowledge over time. Boxes with dashed outlines indicate hidden nodes in the graph while boxes with solid outlines represent observable nodes in the graph.

A.1.1 Formulation

The original knowledge tracing technique (Corbett and Anderson, 1994) used a simple dynamic Bayesian network (DBN). An example of a DBN can be seen in Figure A.1. A DBN is a graphical model used to represent sequential data. There are two types of nodes in a DBN: hidden nodes and observable nodes. As the name implies, hidden nodes represent some information that is unknown whereas observable nodes represent information that can be seen. In knowledge tracing, the student’s knowledge state is represented as a hidden node with two possible values, learned or unlearned. The observable nodes of this model are the student’s performance on a given item. At every time step, which in this case would be every time the student answers a question, the student’s knowledge state is either learned or unlearned with a certain probability. These probabilities are calculated using the following formula:

\[ p(L_n) = p(L_{n-1}|e) + (1 - p(L_{n-1}|e)) \times p(T) \]  

(A.1)

In the above equation, \( p(L_n) \) is the probability that the student is in the learned state at time \( n \), \( e \) is the current set of evidence (whether or not the question was correctly answered), and \( p(T) \) is the probability of transitioning from the unlearned state to the learned state. As you can see in the equation, the probability that a student is in a learned state at a given
time is the sum of two separate probabilities. The first of these is the probability that the student was in the learned state in the previous time step given that student's evidence. The second probability used is the probability that the student transitioned into the learned state if he was not in the learned state during the last time step. It is important to note that this model implies that a student can not \textit{forget} once they have entered the learned state. In other words, it is not possible to transition from the learned state back to the unlearned state at any time.

Using equation A.1, it is possible to make predictions about a student’s performance on future questions. At each step, the probability that a student will answer a question correctly can be calculated using the following equation:

$$p(C_i, s) = p(L_{r,s}) \times (1 - p(S_r)) + (1 - p(L_{r,s})) \times p(G_r)$$  \hspace{1cm} (A.2)

Here, $p(C_i, s)$ is the probability that a student, $s$, answers question $i$ correctly. In the second half of the equation, $p(L_{r,s})$ is the probability that the student, $s$, is in the learned state for the rule $r$ that is needed to solve question $i$, $p(S_r)$ is the probability that a student will \textit{slip} (answer incorrectly on an already known topic), and $p(G_r)$ is the probability that a student will \textit{guess} correctly on a topic that is unknown. So, the probability that a student will answer correctly is also the sum of two probabilities. The first of these is the probability that the student has learned the topic times the probability that they do not \textit{slip}. The second of these is the probability that the student has not learned the topic times the probability that the student has correctly guessed the answer. In this model, there are four parameters which must be fit: the guess parameter $p(G_r)$, the slip parameter $p(S_r)$, the initial probability of knowing a skill $p(L_0)$, and the transition probability $p(T)$.

\textbf{A.1.2 Extensions and Examples}

One problem that was soon discovered with the knowledge tracing model was that it consistently overestimated student performance. Corbett and Bhatnagar (1997) show that the standard knowledge tracing model will, on average, overestimate student performance by 8%. They claim that students are learning suboptimal rules. Corbett and Bhatnagar account for this behavior by adding a third state to this model. The new states of student knowledge are: unlearned, learned incorrectly, and learned correctly. To actually make predictions, Corbett and Bhatnagar included an additive term that is proportional to how
Figure A.2 A DBN showing the initial student knowledge node added in. In the model put forth by Pardos and Heffernan, there would be an initial knowledge node for each student that could be modeled.

well the student is able to acquire ideal rules. They showed that this extension to the framework essentially eliminated the overestimation that they had been seeing.

Another issue with the simple model presented above is that it requires that there only be one KC associated with each task. Therefore, observing student performance on a given task only updates the probability of a single KC being in the learned state. Conati et al. (2002) relax this restriction with the student model in the Andes system (Vanlehn et al., 2005). In other words, one task can have more than one parent node in the Bayesian network. In this system, credit assignment is done to determine which KC is most likely to have contributed to the performance of the student. This is achieved by examining rule prior probabilities. If the prior probability of one concept is much higher than the other, then it will receive most of the credit for the solution through Bayesian update rules.

This model also suffers from the identifiability problem (Beck, 2007). This occurs when the same training data can be fit equally well by different parameter values. This can have very noticeable effects on the generalizability of the model. Beck (2007) offers a solution using Dirichlet priors to guide parameter values towards their means in order to raise the probability that these parameters all converge to the same maximum.

Baker et al. (2008) claim that this introduces a new problem: model degeneracy. Model
degeneracy occurs when parameter values cause the model to violate its conceptual foundations (such as students being more likely to answer correctly if they do not know a skill). Baker et al. address these issues by adding contextualized estimations of the guess and slip parameters. Recall that in the original model these parameters remained constant across all observations. In this model, they examine logs of student performances and determine whether each response was a slip or a guess by examining that student’s future performance. They then construct models through machine learning that will extract rules about when a guess or a slip is likely to occur. They show that this method produces models that are less vulnerable to model degeneracy while being comparable to previous approaches in addressing the identifiability problem.

Pardos and Heffernan (2010) introduce a prior probability over student knowledge in order to add individualization to the model. This way, the model will be able to take into account individual differences between students. This was done by adding in student prior knowledge as the first node in the network. An example of this network can be seen in Figure A.2. They showed that the best way to initialize this parameter was to use the student’s average performance on all observed questions except for the one being predicted and then let this value be updated during parameter fitting.

Recently, the knowledge tracing model was applied to the interactive narrative-based learning environment Crystal Island. To accomplish this, Rowe and Lester (2010) had to identify which in-game behaviors were most indicative of student knowledge and construct the DBN using this knowledge. They showed that using this technique to model student knowledge in a serious game had promise as it performed better than a random baseline classifier for several knowledge thresholds.

A.2 Performance Factor Analysis

Originally introduced as an alternative to knowledge tracing, performance factor analysis (PFA) addresses some of the problems that are present in the knowledge tracing model. The main problem that PFA addresses is that of a task requiring multiple KCs to complete. Since PFA involves a sum over all contributing KCs, it is able to handle multiple KCs contributing to student performance. As with knowledge tracing, PFA requires knowledge of the relationship between individual questions and KCs. Since this must be known beforehand, this technique would be classified as a knowledge engineering approach using the taxonomy that I used
for describing player models.

A.2.1 Formulation

PFA (Pavlik et al., 2009) is based on a technique for educational data mining known as learning factors analysis (LFA) (Cen et al., 2006). In order to better understand PFA, I will first provide an overview of LFA.

The LFA model decomposes learning into three factors that describe different aspects of learning and KCs: the student’s ability, the KC’s easiness, and the KC’s learning rate. Formally, the LFA model is given by the equation below

\[ m(s, j \in k, n) = \alpha_s + \sum_{j \in k} (\beta_j + \gamma_j n_{s,j}) \]  

(A.3)

The formula \( m \) represents the accumulated learning for a student \( s \) using a KC \( j \) from the total set of KCs \( k \). In the above equation, \( \alpha \) is used to encode student ability, \( \beta \) is used to represent the easiness of a KC, and \( n_{s,j} \) is used to determine the weight of prior observations since \( \gamma \) is added for each observation. In order to convert these \( m \) values into predictions of probability, one must use the following equation:

\[ p(m) = \frac{1}{1 + e^{-m}} \]  

(A.4)

For each task that will be presented to the student, the KCs associated with that question are stored in a Q-matrix. This is used to determine the KCs that best describe each task based on observed data.

The main issue with LFA is that it does not take into account student performance beyond the frequency of prior observations. In order to account for this, Pavlik et al. (2009) extended the LFA model to be appropriate for predicting student performance. They did this by, first, eliminating the \( \alpha \) parameter since student ability is typically not known beforehand in an ITS. They also replace the \( n \) variable with two parameters, \( c \) and \( f \), which track the student’s correct responses and incorrect responses respectively. These two variables are scaled by the variables \( \gamma \) and \( \rho \). The resulting equation is given by

\[ m(s, j \in KC_s, c, f) = \sum_{j \in KC_s} (\beta_j + \gamma_j c_{s,j} + \rho_j f_{s,j}) \]  

(A.5)

Equation A.4 is still used to convert these \( m \) values into probability predictions. In this
model, there are parameters that are fit using logistic regression to create a model that best explains the training data. These parameters are $\beta, \gamma, \alpha$, and/or $\rho$.

### A.2.2 Extensions and Examples

One of the main issues with the PFA model is that, while it does take past successes and failures into account, it does not take into account the order that these successes and failures occurred in. Gong et al. (2011) extended this framework to include a decay factor which would decrease the effect that older questions had on current student knowledge. They showed that this version of PFA had a higher prediction accuracy than the original framework by a significant margin.

Another issue with the framework as it was originally formulated is that it assumes that the context in which a KC is learned has no effect on whether or not the concept is learned. In other words, mastery of two different tasks that consist of the same two KCs will result in the same amount of perceived learning for these two KCs. Pavlik et al. (2011) propose a new way of constructing the Q-matrix to help determine which KCs are most likely to be learned at a given time. In the original formulation, KCs control an entire column of the Q-matrix and so when they are fit, they are fit for all tasks. In the proposed method, each cell of the Q-matrix is fit. As a result of this, KC associations are found for specific tasks rather than finding KC associations across all tasks. Pavlik et al. tested this method on a least common multiples (LCM) skills dataset and compared the Q-matrix produced by their method against the Q-matrix produced by PFA. Pavlik et al. found that PFA did not associate any KCs to tasks in the dataset whereas their method found several associations.

Chi et al. (2011) explored adding in support for other types of instructional intervention into the PFA model. Their model, which they have named the instructional factors analysis model (IFM), also receives input about what they call tell actions. In the ITS that they study, a tell action is an instructional intervention where the system tells the student what steps to take next. Previous iterations of PFA do not take these actions into account since they do not have an immediately observable effect on the student. To account for this, Chi et al. add in a variable to account for the number of tell actions that the student has previously received on a given KC to equation A.5. They showed that IFM outperformed the standard implementation of PFA on data collected by a natural language physics tutor named Cordillera.
A.3 Matrix Factorization

Recently, matrix factorization has been used to predict student performance in tutoring systems. Matrix factorization is a technique that was originally used in recommender systems. The goal of a recommender system is to either recommend new items to a user based on that user’s history, or to predict how a user will rate an item based on how they rated items in the past. Using this, Thai-Nghe et al. (2011a) claimed that the problem of predicting student performance is similar, computationally speaking, to that of predicting user ratings. With this in mind, they converted the matrix factorization algorithm into a form that makes it fit for predicting student performance. This technique for student knowledge modeling is a data-driven technique since the only prior knowledge required is the number of latent factors. This is a value that can be optimized to achieve the best performance if need be.

A.3.1 Formulation

Matrix factorization is the task of approximating a matrix $X$ as the product of two smaller matrices $W$ and $H$. Here, $W \in \mathbb{R}^{U \times K}$ is a matrix containing the latent factors that describe each student and $H \in \mathbb{R}^{I \times K}$ is a matrix containing the latent factors that describe each task. In this framework, the performance of student $u$ on task $i$ is predicted using the following equation:

$$\hat{p}_{ui} = \sum_{k=1}^{K} w_{uk} h_{ik} = (WH^T)$$  \quad (A.6)

To use this equation, however, I must find optimal values for the elements of the matrices $W$ and $H$. These values are typically found using gradient descent in order to minimize
some error function. An example of this technique will be done with the aid of Figure A.3. Let’s assume that the matrices $W$ and $H$ seen in Figure A.3 were obtained after training for $K = 2$ latent factors. If I wanted to predict the performance of Student 2 on Task 2 (which are shown in bold and italics in the figure), I would simply do the following:

$$\hat{p}_{ui} = (WH^T) = 0.11 \times 0.94 + 0.32 \times 0.63 = 0.305$$

In this example, the student’s predicted performance is 0.305. Since performance is typically reported as 0 if the student incorrectly answers a question and a 1 if the student correctly answers a question, I can conclude that it is unlikely that the student will correctly answer the question.

As it is, this model does not take into account student bias, the ability of the student, or the task bias, the difficulty of the task. To incorporate these into this model, three new variables are introduced: the global average performance $\mu$, the student bias $b_u$, and the task bias $b_i$. The new equation for prediction becomes

$$\hat{p}_{ui} = \mu + b_u + b_i + \sum_{k=1}^{K} w_{uk} h_{ik} \quad (A.7)$$

In the above equation, $b_u$ is the average amount that the performance of student $u$ deviates from $\mu$ and $b_i$ is the average amount that the performance of task $i$ deviates from $\mu$. When training this model, these values are updated using the predictions made from the current values of $W$ and $H$.

Now the only thing the model lacks is a way to describe time. To do this, Thai-Nghe et al. (2011c) used a technique called tensor factorization. Tensor factorization is a generalized form of matrix factorization. Specifically, a new matrix $Q$ is introduced which describes the context of a task (the time). After adding this extra component to the model, the equation for prediction becomes

$$\hat{p}_{ui} = \mu + b_u + b_i + \left( \sum_{k=1}^{K} w_{ik} h_k \phi_k \right) \quad (A.8)$$

$$\phi_k = \frac{\sum_{t=(T-T_{max}+1)}^{T} q_k(t)}{T_{max}} \quad (A.9)$$

In the above equations, $q$ is a vector containing the $K$ latent factors that affect time, and
$\phi$ is the average value of $q_k$ looking $T_{max}$ steps into the past. In other words, this model will use the average performance over the last $T_{max}$ tasks to predict the student’s performance on the current task. It is interesting to note that this model does not have explicit parameters that account for guess and slip actions. This is because they are assumed to be accounted for as latent factors that describe the student and the tasks they perform.

### A.3.2 Extensions and Examples

An issue with matrix factorization that was pointed out by Thai-Nghe et al. (2011b) is that it only considers the relationship between the student and tasks/skills. They point out that there are several other relationships that could be considered in an ITS, such as the relationship between tasks and the skills they require. To address this, they extend the matrix factorization method by using multi-relational matrix factorization (MRMF) (Lippert et al., 2008). MRMF is a generalization of matrix factorization in which multiple relationships and entities (such as students or tasks) can be used. Thai-Nghe et al. evaluated this method on the KDD cup 2010 educational data mining dataset for algebra and bridge to algebra.¹ They showed that using MRMF results in an overall lower root mean squared error than both normal matrix factorization as well as knowledge tracing.

Recently, Zook et al. (2012) used tensor factorization to create models of player expertise in a skill-based mission game. In this work, they described missions in terms of what skills are necessary to complete them. Using tensor factorization, Zook et al. were able to predict how player performance would change over time. They hypothesized that they can use this predicted player performance to customize the missions that a player would experience. What is interesting about this is that they could customize missions in order to produce a desired performance curve. This is also an example of a student model from an ITS being used outside of the ITS field. In this case, it has been applied to a game meant solely for entertainment.