Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment

Trigram Timmies and Bayesian Johnnies: Probabilistic Models of Personality in *Dominion*

Kevin Gold
Rochester Institute of Technology

Abstract

Probabilistic models were fit to logs of player actions in the card game *Dominion* in an attempt to find evidence of personality types that could be used to classify player behavior as well as generate probabilistic bot behavior. Expectation Maximization seeded with players' self-assessments for their motivations was run for two different model types – Naive Bayes and a trigram model – to uncover three clusters each. For both model structures, most players were classified as belonging to a single large cluster that combined the goals of splashy plays, clever combos, and effective play, cross-cutting the original categories – a cautionary tale for research that assumes players can be classified into one category or another. However, subjects qualitatively report that the different model structures play very differently, with the Naive Bayes model more creatively combining cards.

Introduction

Personality: players have it, and we would like artificial agents in games to have it, too. Player personality can come through in terms of which strategies or play styles they prefer to engage in; the PaSSAGE system, for example, categorizes roleplaying gamers as Fighters, Storytellers, Method Actors, Tacticians, and Power Gamers (Thue, Bulitko, and Spetch 2008), while Bartle made the observation that massively multiplayer game players tend to fall into the categories of Achievers, Explorers, Socializers, and Killers (Bartle 2003). Ideally, if we could detect a player's play style or motivation, it would be possible to give the player more of what the player wants, and less of what the player prefers not to deal with. In multiplayer matches, we may even want to pair players off who share values about what makes a good game. A player who enjoys creative, offbeat approaches to the game may not want to play against a player who uses all the latest optimal strategies off the Internet, and vice versa; the creative player may feel that the effective player's strategies are cheap, while the effective player may feel that the creative player provides no challenge.

When it comes to AIs in games, we know many ways of making them optimal, and few ways to make them play with personality or in a "fun" way. Methods such as Expectiminimax or game theoretic approaches can solve games optimally given enough processing power, but have no way to degrade gracefully besides reduced lookahead, which does not exactly equate to personality. In machine learning, usually there is a right and a wrong answer, and all agents are herded toward rightness. Approaches to personality in games are typically hardcoded and not meant to coexist with competitive, complex behavior; the makers of Microsoft's *Bicycle Texas Hold 'em* characterized poker AIs by how "loose" versus "tight" and how "aggressive" versus "passive" the AIs were in their bets; this conveyed personality, but the AIs were predictable and easily beaten (Ellinger 2008).

This work explores using probabilistic models to mine personalities and unpredictable but sensible play from existing player data. By measuring conditional probabilities of play from game logs, models can be created that particularly match segments of the player population in their frequencies of choosing particular plays. The models can serve several purposes. Run on a particular player, they can be used to classify the player according to type, and then predict the probabilities of the player's next plays. Run as a bot, the model can generate behaviors matching observed player frequencies; through careful choice of conditional dependence relationships, the bot behavior can be sensibly predicated on its previous plays while remaining unpredictable. Finally, the exact categories of players need not be fixed ahead of time; using Expectation Maximization, the models can drift to the clusters actually present in the data, rather than assuming a player ontology that may simply be false.

Building models that can be applied to both bots and players is a novel approach that comes with several potential advantages over other techniques. First, the play that is generated is not optimal play that has been subsequently brain damaged, but matches observed player behavior; it therefore degrades more gracefully, insofar as it will always match some group of players' rough frequencies of play, and ideally generates more humanlike play. Second, the probabilities are generally inspectable, which allows the personalities generated to give some insight into how actual players play the game, and why particular plays of the AI are made. Third, the behavior created can be unpredictable, since it selects actions from a conditional probability dis-

tribution rather than always choosing the optimal play; this should make playing against bots derived from the models more fun.

The particular game examined here is *Dominion*, a popular fantasy card game that is reminiscent of *Magic: the Gathering*. As in *Magic*, each card behaves differently, and can contain rules text that changes the way the game is played; "Masquerade," for example, has players passing cards to their left, while "Possession" allows a player to take control of another player's turn, and "Stash" is a card that can be placed anywhere in the deck after shuffling. Unlike *Magic*, *Dominion* is not a collectible card game; instead, players attempt to assemble decks of clever combos as the game progresses, "buying" cards for their decks by playing cards from their hands that allow them to acquire better cards (Figure 1). Coding the AI behavior for each of the 138 cards available for the game was beyond the scope of this paper, but the heart of the game is in what players choose to "buy" to put in their decks, and this is the behavior we will be concerned with for the rest of the paper. The game is interesting in that many cards are not useful by themselves, but only in conjunction with others; while "King's Court," which plays any other card in the hand three times in a row, is one of the most powerful cards in the game, an AI that chose to buy only King's Court cards would be in dire straits. An online server exists for the game that helpfully logs all games, of which there are currently (mid-May 2011) over 10,000 being played a day; this paper limits itself to using only six days of data for training, and a seventh for evaluation.

Here, we compare two probabilistic models that can be trained with these logs of play: a Markov model that conditions on the last card bought as well as the last card bought at the same price; and a Naive Bayes like model that conditions on how likely each buy is given the presence or absence of each available card in the player's deck. Within each model, Expectation Maximization was run to create three clusters of players with different parameters; these models were initialized based on players' statements about what they liked about the game, to see whether these differences persisted in creating clusters based on preferred play styles. The player motivations considered were the categories suggested by *Magic* lead designer Mark Rosewater for why people liked collectible card games: "Timmy," who plays to experience the thrill of big, splashy plays; "Johnny," who enjoys the games because of the creativity in coming up with new combos; and "Spike," who plays simply to win (Rosewater 2002). Though the clusters were initialized with the ground truth of real players' self-assessed motivations for playing, the clusters that emerged painted a rather different picture in *Dominion*.

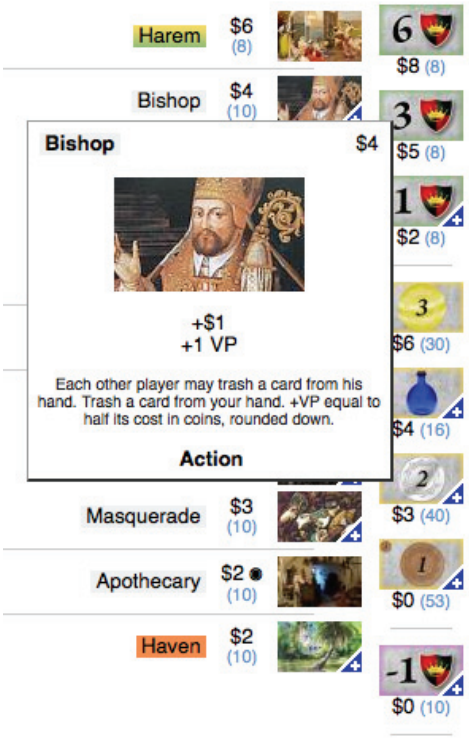

Figure 1: A screenshot of the available buy screen, from dominion.isotropic.org, including help text for the Bishop card. (Five more actions are obscured by the help text.)

Background

Related work

The idea of building Bayesian network personality models is novel to this work, but it builds on related work in probabilistic reasoning about goals, player goal and play style detection, and models of agent personality.

One of the more direct inspirations for this work was a model for recognizing player goals in multi-user dungeons, from their locations and actions (Albrecht, Zukerman, and Nicholson 1998). The model was able to recognize which quest the player was on with a high degree of accuracy, and to a more limited extent, predict the player's next action and location. A variant on such a system for real-time, action-oriented games used input-output hidden Markov models to predict a player's current goal (Gold 2010).

Approaches to personality in games have generally taken the approach of modeling personality as a sliding scale on several attributes; interestingly, these approaches often end up deciding to set the scale all the way in one direction or another, to more clearly get across the concept that a personality is present (Ellinger 2008; Evans 2009). Some approaches also direct non-player character behavior in a way that more closely matches the player's perceived preferences (Yannakakis and Hallam 2004).

Two primary approaches that have been used to fit AI models to human behavior in games are case-based reasoning (Laviers et al. 2009) and plan network learning (Lesh and Etzioni 1996). The present work differs from case-based reasoning in that this work is probabilistic, and selects actions according to a distribution based on the amalgamation of many games, with similarity to actual games observed only in the expectation. Jeff Orkin's *The Restaurant Game* has been used to train AIs to behave sensibly in roleplay-

ing a restaurant interaction, automatically clustering player behavior into "sensible" and "not sensible" based on the geometric mean of the Markov transition probabilities in a plan network learned from online play (Orkin and Roy 2007). This work draws inspiration from *The Restaurant Game* and presents two ways of conditioning on slightly more information, as well as a more general approach for clustering behaviors. *Starcraft* build order strategies have been studied in the context of performing machine learning to classify hand-labeled strategies, including a variety of machine learning algorithms such as C4.5, nearest neighbors, and exemplar-based methods (Weber and Mateas 2009). The present work is a step toward automatically generating the kinds of clusters that were hand-labeled in that work, and also shows how the same model used for recognition can be run "in the opposite direction" to produce interesting non-deterministic behavior.

It is also known that Electronic Arts logs play probabilities in their sports games, and uses these to drive AI behavior (Guevara 2010); but no smoothing or generalization occurs in that work, leaving their data set somewhat sparser than it could be.

Dominion and isotropic.org

Dominion is a card game in which each player assembles his or her own deck over the course of the game, in the hopes of producing a deck that has good synergy and can ultimately rack up the most points. Cards fall into three categories: action cards, which are the heart of the game and do whatever they say in their text; treasure cards, which allow the player to acquire more cards for their decks; and victory cards, which are simply worth points at the end of the game. A player normally may play only one action card and "buy" one new card for their deck per turn, but action cards can generally allow the player to exceed these limits; in fact, a player may end up having a very lengthy turn if actions are played to get more actions, some of which are used to draw more cards, which themselves may be actions that increase the number of actions, and so on. The game supports 2 to 4 players.

The action cards are at the heart of the game, and a set of 10 action cards available for purchase (out of over 138 possible at the time of this writing) is different every time (Figure 1). Most action cards do more than one thing; "Witch," for example, allows the player to draw 2 cards and also places a useless Curse card worth -1 point into each other player's deck, making their draws less useful. Some cards do nothing by themselves, but are meant to used in conjunction with other cards; "King's Court" takes any other action in the player's hand and plays it three times.

Fans of the game have set up an online server (dominion.isotropic.org) which logs all the games played, including which cards were available, the usernames of players, and turn by turn plays and buys. This particular study will only make use of 6 days of play – May 1 to 6, 2011 – for its training data, and a 7th for some predictive evaluation. The six-day data set contains 62,045 game logs from over 6000 players, while the seventh day contained 8,294 logs for evaluation. The logs available on the site cover several months of play, and are available for anonymous download.

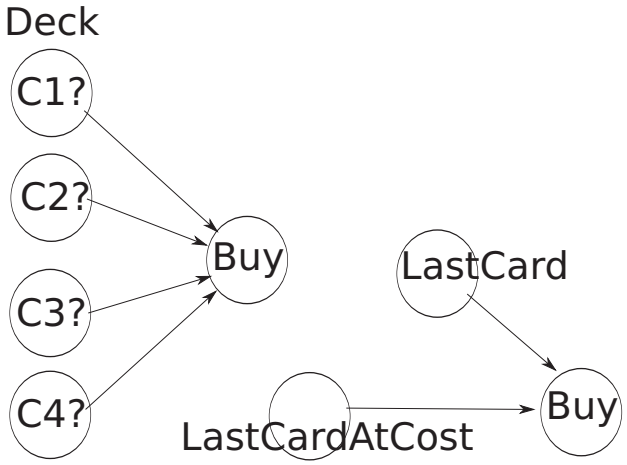

Figure 2: The two graphical models used to determine card buy probabilities. The Naive Bayes model, left, conditions on the presence or absence of cards in the deck, while the trigram model conditions only on the most recent buy and the most recent at the same cost.

Models and Expectation Maximization

A Bayesian network is a concise means of expressing the probability that an event occurs, given the occurrence or non-occurrence of related events; see (Hyde 2008) for an introduction to their game-related uses. A node in a Bayesian network corresponds to the outcome of a particular event; a "conditional probability table" for each node expresses how often the event occurs given some other event.

In this particular work, the models represent player buy patterns over the course of the game. Two different models were evaluated: a Markov-like "trigram" model that selects each card based on the last buy as well as the last buy at the same cost; and a dynamic "Naive Bayes" model that selected cards based on which of the cards in the pool were already in the player's deck. The two models' structures are summarized in Figure 2.

Once trained, these models can be used for three purposes: producing behavior with the frequencies of an actual group of players; classifying a player to a cluster of players with similar buy frequencies; or predicting a player's next buy, using the frequencies of that player's cluster. Note that all buys are public, as are all game logs on the server, so an AI opponent always has the necessary information to both classify a player and predict the player's next buy.

Dynamic Naive Bayes model

A player's decision about which card to buy next for the deck is contingent on the current content of the player's deck, the selection of cards available for purchase, and the money available. The content of the player's deck is modeled here as a set of Boolean variables $C_1, C_2, \ldots, C_N$, representing for each card available for purchase whether the player has al-

ready bought it and put it in the deck. The goal here is recovering the quantity

$$\beta_b = P(Buy = b | C_1, C_2, \ldots, C_N, Money = m) \quad (1)$$

with the β_b simply being a convenient shorthand for the equations to follow. To choose a card semi-randomly in a player-like manner, we would like this probability to be as close as possible to some actual group of players' behavior, without needing to specify an exponential number of conditional probabilities.

Invoking Bayes' rule to turn around the conditional probabilities results in:

$$\beta_b \propto P(C_1, \ldots, C_N, Money | Buy = b) P(Buy = b) \quad (2)$$

Now we make the Naive Bayes conditional independence assumption – equivalent here to assuming that the cards in the deck will all "hang together" purely by virtue of this next card – and the equation becomes

$$\beta_b \propto P(b) P(Money|b) \prod_i P(C_i|b) \quad (3)$$

We will make the simplifying assumption that players will always buy one of the most expensive cards available, so $P(Money|Buy)$ is quietly dropped from the equation; though in principle, we might condition on similar terms for as many independent factors as care to.

$$\beta_b \propto P(b) \prod_i P(C_i|b) \quad (4)$$

The final equation is just the product of $P(C_i|Buy = b)$ for all the possible cards in the deck and $P(Buy = b)$ for the card in question. These can be thought of as representing how often the card is comboed with each other card in the deck, as well as how popular the card is by itself. Both can be easily obtained from the game logs by counting how often each card was in the deck when a particular *Buy* was purchased, and comparing it to the total number of times the card was available for purchase when *Buy* was bought. A similar pair of counts can track how often a particular card was bought when it was available for purchase. Note that a card that is available for purchase but not in the player's deck can still affect the current buy, because the probability that a card is bought given another card is *not* in the deck (but is in the supply) is included in the product for each card in the supply. Cards can therefore encourage or discourage the purchase of other cards simply by virtue of being in the supply. (Cards not available for purchase in a particular game cannot affect the probability.)

This results in a relatively concise model, but it can be the case that model likelihood calculations can return *NaN* without some smoothing for unobserved events. Thus, all the observation counts are initialized to pseudocounts of 1, and all the "possible buy" counts are initialized to 2, to give starting probabilities of 1/2 for all cards in the absence of evidence.

Given a set of players to measure probabilities from, calculating all of these parameters requires only a single linear time pass through the data, keeping track of which cards were in a player's deck at each time in a hash table. (The Expectation Maximization algorithm described later is only of interest if we are trying to detect clusters of players.)

Once a model is built, it can be used for three purposes. First, to find a player's most likely buy at each time step given a player model, equation 4 is calculated for each buy of the most expensive price that the player can afford, and the card with the highest number is chosen.

Second, to find the probability of each buy for the purpose of deciding behavior semirandomly, the value in equation 4 is scaled by the sum of all such values for a particular buy price, forcing the probabilities to sum to 1 in typical Bayesian fashion:

$$\beta_b = \frac{P(b) \prod_i P(C_i|b)}{\sum_j \beta_j} \quad (5)$$

Third, we can compare multiple models to find the model that best fits a player. Bayes' Rule suggests that the probability that the probability the player fits the model given their actions is proportional to the probability of their actions given the model. This is just the product of the buy probabilities of equation 5 over all recorded buy sequences for that player; the model with the highest likelihood wins. In practice, of course, multiplying small probabilities results in underflow, so the logs of the probabilities are added instead. Deciding which model best fits a player is also a linear time operation.

Trigram model

As another Bayesian model, the trigram model worked very similarly to the Naive Bayes model, but its buy choices were only conditioned on two factors: the previous buy, and the previous buy at the current amount of money. (This is not the same as conditioning on the last two cards, which would be a more precise analogy with "trigrams" in language processing; but because we are counting triplets, the term "trigram" seems as good a shorthand as anything.) $P(Buy = b | LastCard, LastCardAtCost)$ was measured for each triplet of cards, and for any particular buy decision, the available choices had their probabilities normalized to sum to 1. Further details are omitted for space.

Expectation maximization

Expectation Maximization can be used to define clusters of players without knowing ahead of time what those clusters are, and refine their models even as players shift category. The process consists of alternating between an "Expectation" step, in which the log likelihood that each player belongs to each model is calculated, and a "Maximization" step, in which the conditional probabilities of each model are recomputed based on the new reassignments of players to categories, to maximize the likelihood of the observed behavior (though here, the probabilities that maximize the likelihoods are just the observed frequencies of actions).

Though some EM methods weight the evidence by the probability that a player represents one category or another, the model likelihoods calculated for each player here are orders of magnitude different, making contributions from the less likely models negligible, so we simply count a player as belonging to the most likely category.

Player categories are completely reassigned during Expectation, and model probabilities are completely recomputed during Maximization. The initial seed was players' self-descriptions for their motivations for playing (see below). EM was run until it qualitatively appeared to converge, when the number of players shifting category did not change much between trials.

Experiment

Methods

Eleven Dominion players were recruited online and asked to choose which of the following descriptions best described what they enjoyed about *Dominion*:

> *Timmy*: I like it when cool things happen, e.g. because they're built into the cards (Possession), or because there was a long chain of action splitters that gets you a lot of money, or Masquerade ended up doing something weird.
>
> *Johnny*: I like it when I can find a quirky, unique way to win, like buying all the curses to end the game, or finding a weird use for Pearl Diver.
>
> *Spike*: I like using my knowledge of what cards are better than others and playing strategically to win.

These players' choice of self-description were then used as the seeds for Expectation Maximization, the first pass of which consisted of setting the three models' parameters based on the observed action frequencies of these three categories of players. There were five self-identified Spikes, three self-identified Johnnies, and three self-identified Timmies.

The models were then trained on all player data between May 1 and May 6, using Expectation Maximization to adjust the player categories. EM was run until it appeared to converge, by virtue of the number of players changing category reaching a steady state. This took fourteen passes for the Naive Bayes model, with about 50 players still changing category, and five for the trigram model, with only 1 still changing category.

Results

Both model structures, trigram and Naive Bayes, resulted in all of the original self-classifying Dominion players being reclassified into a single category over the course of EM. In both models, this category appeared to combine some elements of the thrill-seeking "Timmy" personality and some elements of victory-seeking "Spike." In both model structures, the probability of buying "action splitters" that allowed more actions and the probabilities of buying generally high-quality cards were much higher than the other two models, which received only a fraction of the players (roughly 500 each, compared to 5500 in the main category). We call this primary cluster "MainlineCombo" for both models. In the trigram model, the buying preferences looked drastically different for this category than the others; of the top four cards by marginalized conditional probability, three provided more actions. By contrast, the other two models took a more plodding approach to victory, buying Victory cards at earlier opportunities instead of building up to giant plays in the late game; we call these clusters "NoCombo1" and "NoCombo2" in Figure 3, since inspection of buy probabilities did not suggest qualitative differences between the two. Inspecting buy probabilities for the two less popular Naive Bayes models suggested that one took a more slow and steady approach to victory, while the other was a more quirky model valuing alternate win conditions such as Vineyard highly; these are called "BigMoney" and "AlternateWin" in Figure 3. Thus, in each case, the EM procedure appeared to uncover a single "mainstream" strategy and two ways to diverge from the mainstream that were unrelated to the original seeds.

Model/Cluster	Player Type		
Trigram	NC1	NC2	MC
NoCombo1	**0.69**	0.67	0.68
NoCombo2	0.68	**0.71**	0.70
MainlineCombo	0.70	0.70	**0.74**
Naive Bayes	BM	AW	MC
BigMoney	**0.70**	0.68	0.65
AlternateWin	0.67	**0.73**	0.67
MainlineCombo	0.66	0.71	**0.73**
Random priciest	0.51		
Priciest in-deck	0.61		

Figure 3: Cross-model predictive accuracy for the two model structures, and two baseline predictive strategies. The two smaller clusters are different for each model, and the player type headers are abbreviations for the cluster names on the left. On the whole, the model structures perform comparably in prediction despite their different feels. Using the correct cluster to predict a player produces more noticeable differences for the Naive Bayes model.

To get an idea of how different the models were, the models' next card predictions were carried out for each logged game on May 7. Players on that day were first classified into their best-fit models using model likelihood and the parameters learned from the first 6 days, and all three clusters for both approaches were used to predict each player's next buy at every stage of the game. This test set consisted of 8,294 games. These were compared to the baselines of buying a random card from the most expensive cards the player can afford, and a similar baseline that deterministically picks a card already in the player's deck when such a card exists. The results are shown in Figure 3.

Despite having very similar accuracy at predicting human behavior, subjects reported that the models felt very different in practice. By virtue of predicating its actions only on the most recent action and its previous buy, the trigram model gave the impression of a rather impulsive player, gravitating toward good cards, but with no real plan. The Naive Bayes

model, on the other hand, constantly surprised with clever and offbeat combinations of cards, by virtue of its learning the conditional probabilities between cards and having a way for cards in the deck and cards in the supply to affect buying decisions throughout the game. For example, it picked up on the three card combination of Potion-Mine-Bank, an uncommon and interesting play. Quantitative evaluation of the claim that players feel the structures have different personalities has not yet been performed.

Conclusions

On the whole, the Naive Bayes model performed much more intelligently in practice, so it is somewhat surprising that it does no better at predicting human performance than the trigram model. The natural follow-up experiment should be to determine whether EM can assign players to one model *structure* or other based on how well the models capture player behavior. In preliminary experiments, subjects reported that the two structures felt as though they played with very different personalities, so different conditional structures may capture player personality better than card combination preferences. Perhaps all players attempt to play optimally, but pay attention to different information.

While we have characterized the Naive Bayes models as "Big Money" and "Alternate Win," it is important to note that this is on the basis of slightly different conditional probabilities, and even players in these clusters do not always play these strategies. Players may choose to employ particular strategies or not, but such strategies may not actually be *personalities* that characterize player behavior across situations. Experienced players will employ Big Money when it is useful to do so, but there does not appear to be a large cluster of players that are obsessed with the strategy to the exclusion of others. Strategies are not the same thing as player personalities.

Probabilistic models are quite versatile, and this work explores the ways that the same model can be put to multiple uses in game AI – either classifying players, or generating predictions for player behavior, or generating behavior, all from the same model. This is a degree of versatility that is not exhibited by pure classification methods such as feedforward neural networks or SVMs. It was also repeatedly useful over the course of this work to be able to examine the conditional probabilities of the models, to determine how their behavior was put together – again, a feature not readily available to several other methods.

This work did not particularly find evidence for the "Timmy," "Johnny," and "Spike" models of player motivation. Apparently, players can behave very similarly in-game, but for different reasons. It may be that these models are specific to *Magic* or collectible card games in particular; or it may be that they say more about how players view themselves than how they behave in-game. Nevertheless, they may be a useful way to talk about bot personalities.

The fact that the final clusters did not reflect their seeds suggests that future work must be careful to not assume its ontologies. One survey respondent objected to the classification system in the first place, arguing that her reasons for enjoying *Dominion* cross-cut the categories; the EM results bore this out. Despite the success of classification as a data mining approach, generative models and models that are allowed to shift their category systems may show that players do not fit neatly into researchers' preconceived categories.

References

Albrecht, D.; Zukerman, I.; and Nicholson, A. E. 1998. Bayesian models for keyhole plan recognition in an adventure game. *User Modeling and User-Adapted Interaction* 8(1–2):5–47.

Bartle, R. 2003. *Designing Virtual Worlds*. New Riders Games.

Ellinger, B. 2008. Artificial personality: a personal approach to AI. In Rabin, S., ed., *AI Game Programming Wisdom 4*. Boston, MA: Course Technology. 17–26.

Evans, R. 2009. AI challenges in Sims 3. Invited talk, AIIDE.

Gold, K. 2010. Training goal recognition online from low-level inputs in an action-adventure game. In *Proceedings of AIIDE 2010*. Menlo Park, CA: AAAI Press.

Guevara, D. 2010. How AI is applied to commercial games. FLAIRS Invited Talk.

Hyde, D. 2008. Using Bayesian networks to reason about uncertainty. In Rabin, S., ed., *AI Game Programming Wisdom 4*. Boston, MA: Course Technology. 429–442.

Laviers, K.; Sukthankar, G.; Molineaux, M.; and Aha, D. W. 2009. Improving offensive performance through opponent modeling. In *AIIDE 2009*. AAAI Press.

Lesh, N., and Etzioni, O. 1996. Scaling up plan recognition using version spaces and virtual plan libraries. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, Washington.

Orkin, J., and Roy, D. 2007. The Restaurant Game: Learning social behavior and language from thousands of players online. *Journal of Game Development* 3(1):39–60.

Rosewater, M. 2002. Timmy, Johnny, and Spike. "Making Magic" article, www.wizards.com/magic/.

Thue, D.; Bulitko, V.; and Spetch, M. 2008. Player modeling for interactive storytelling: a practical approach. In Rabin, S., ed., *AI Game Programming Wisdom 4*. Boston, MA: Course Technology. 633–646.

Weber, B., and Mateas, M. 2009. A data mining approach to strategy prediction. In *Proceedings of the IEEE Symposium on Intelligence and Games*.

Yannakakis, G. N., and Hallam, J. 2004. Evolving opponents for interesting interactive computer games. In *Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior '04: From Animals to Animats 8*, 499–508. MIT Press.